

TRIANGULATING POINT SETS

Vera Sacristán

Discrete and Algorithmic Geometry
Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

TRIANGULATING POINT SETS

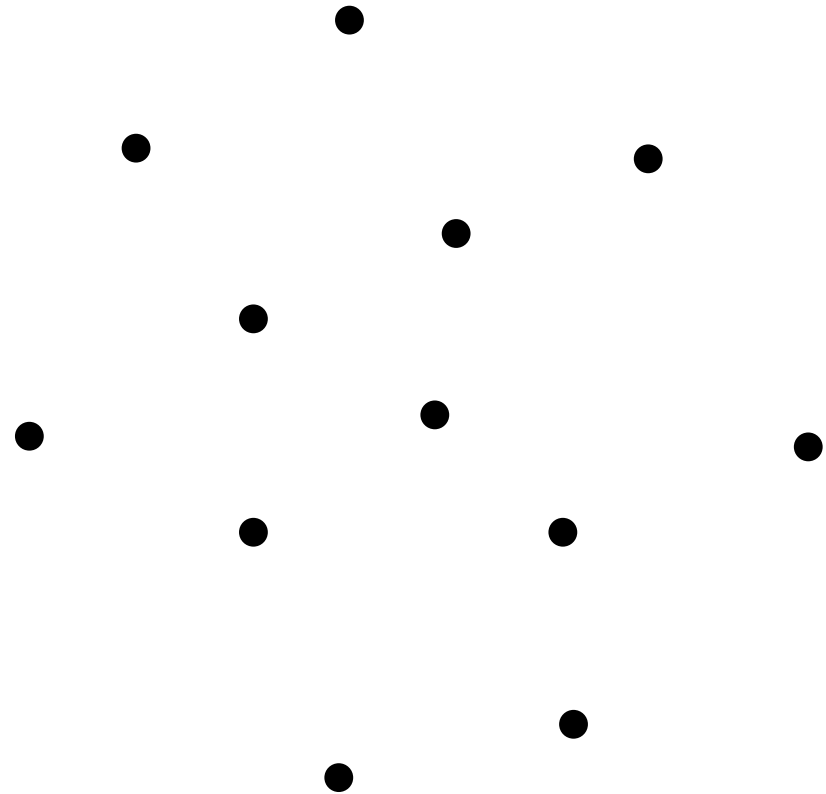
DEFINITION

A triangulation of A set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.

TRIANGULATING POINT SETS

DEFINITION

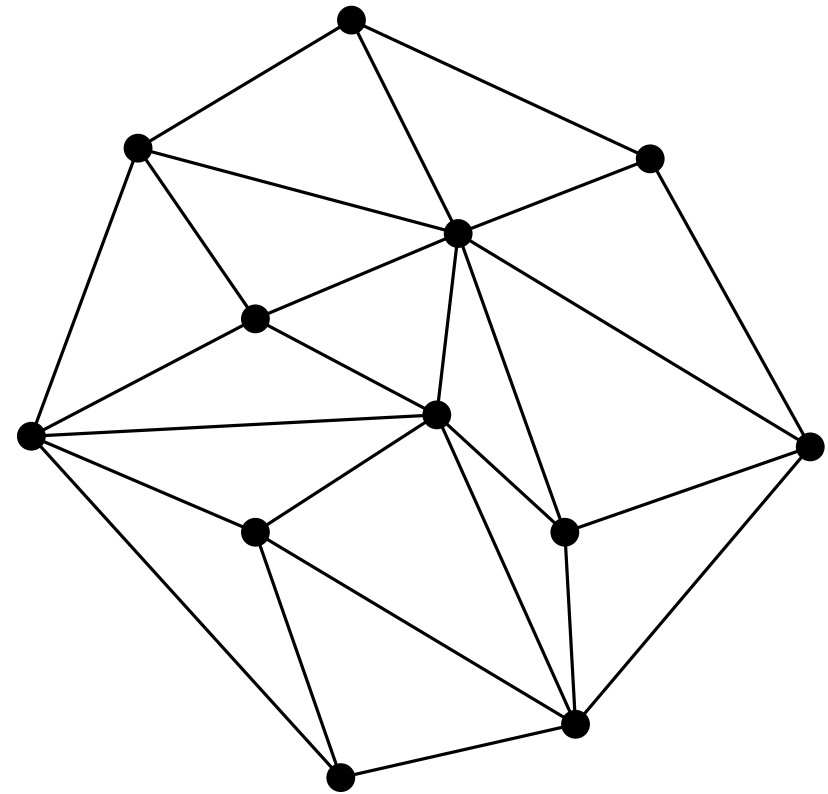
A triangulation of A set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.



TRIANGULATING POINT SETS

DEFINITION

A triangulation of A set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.

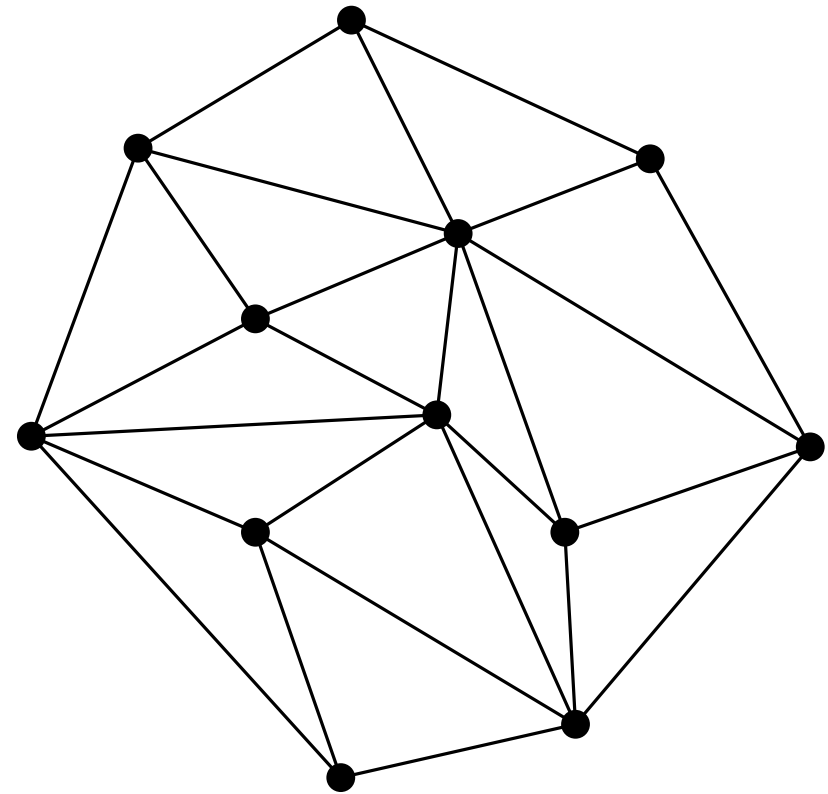


TRIANGULATING POINT SETS

DEFINITION

A triangulation of A set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.

Corollary. All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of P .



TRIANGULATING POINT SETS

DEFINITION

A triangulation of A set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.

Corollary. All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of P .

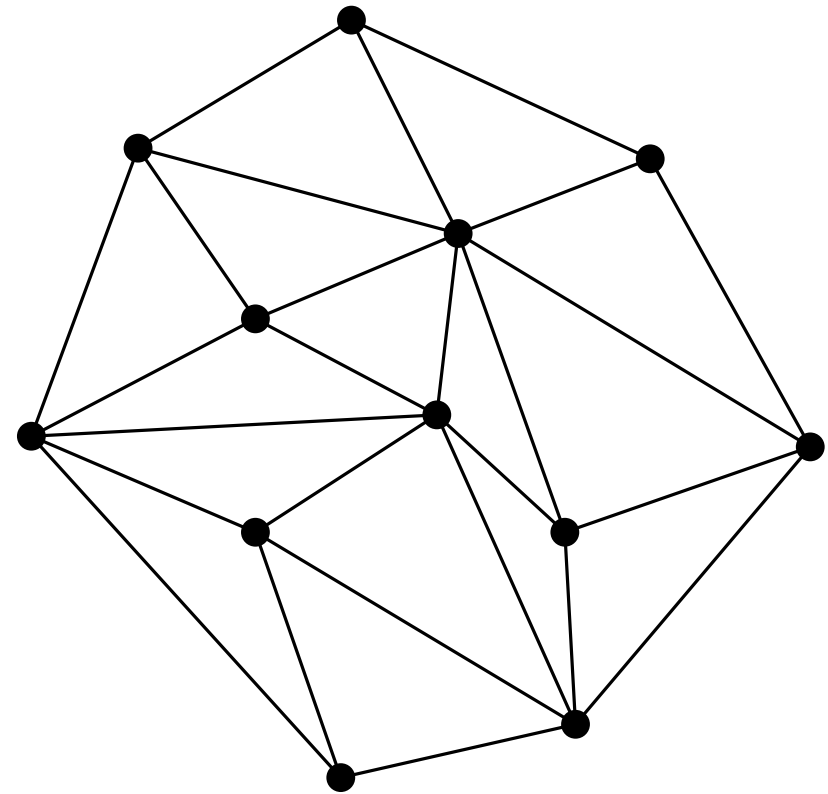
COMPLEXITY

Every triangulation of any set P of n points has:

$$2n - h - 2 \text{ triangles}$$

$$3n - h - 3 \text{ edges}$$

where h is the number of vertices of $ch(P)$.



TRIANGULATING POINT SETS

DEFINITION

A triangulation of a set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.

Corollary. All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of P .

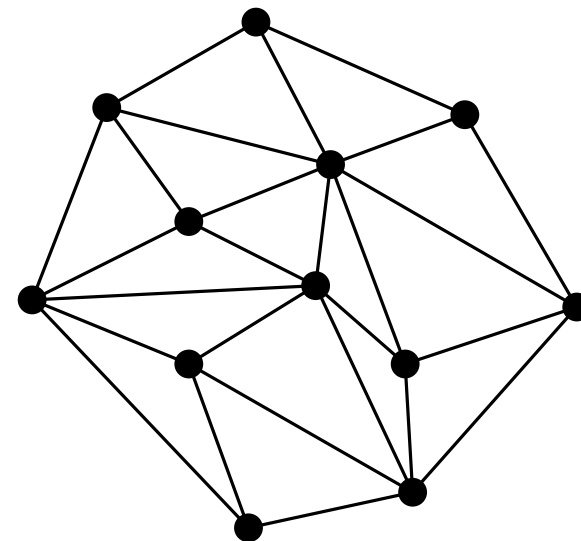
COMPLEXITY

Every triangulation of any set P of n points has:

$$2n - h - 2 \text{ triangles}$$

$$3n - h - 3 \text{ edges}$$

where h is the number of vertices of $ch(P)$.



Proof. Each triangle has exactly 3 edges. Each internal edge belongs to exactly 2 triangles. Each external edge belongs to exactly 1 triangle. Therefore, $3t = 2(e - h) + h = 2e - h$. According to Euler's formula: $n + (t + 1) = v + f = e + 2$.

Combining both equations:

$$e = n + t - 1 \Rightarrow 3e = 3n + 3t - 3 = 3n + 2e - h - 3 \Rightarrow e = 3n - h - 3$$

$$3t = 2e - h = 6n - 2h - 6 - h = 6n - 3h - 6 \Rightarrow t = 2n - h - 2$$

TRIANGULATING POINT SETS

DEFINITION

A triangulation of a set P of n points in the plane is a graph having P as set of vertices which is rectilinear, planar, and maximal in the number of edges.

Corollary. All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of P .

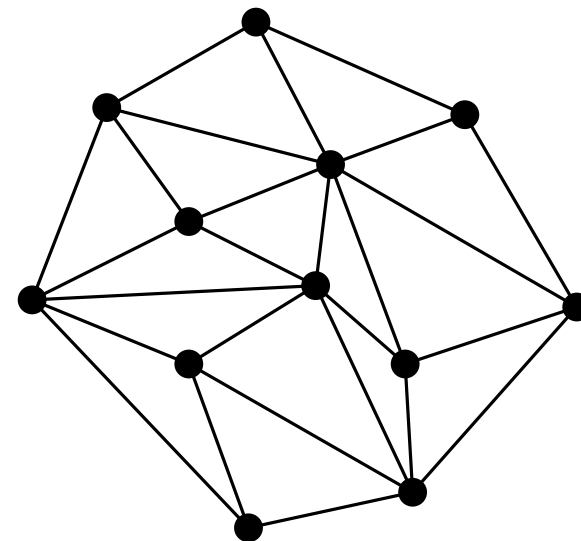
COMPLEXITY

Every triangulation of any set P of n points has:

$$2n - h - 2 \text{ triangles}$$

$$3n - h - 3 \text{ edges}$$

where h is the number of vertices of $ch(P)$.



DEGENERACIES

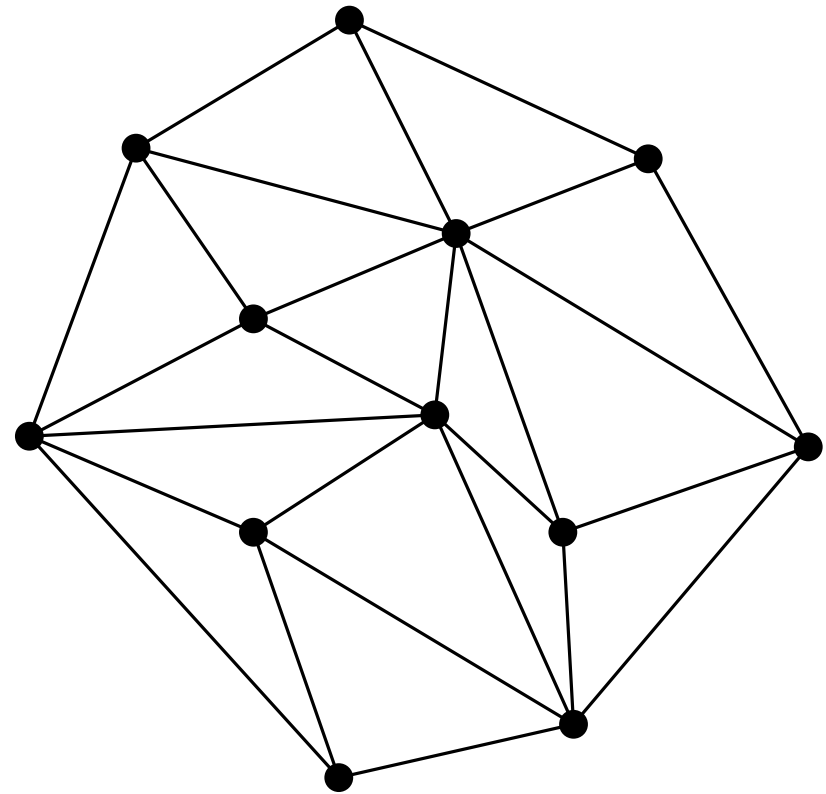
As you may have noticed, we are assuming that the set P does not contain three or more points on a line. The assumption holds along the entire chapter.

TRIANGULATING POINT SETS

DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:

- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.



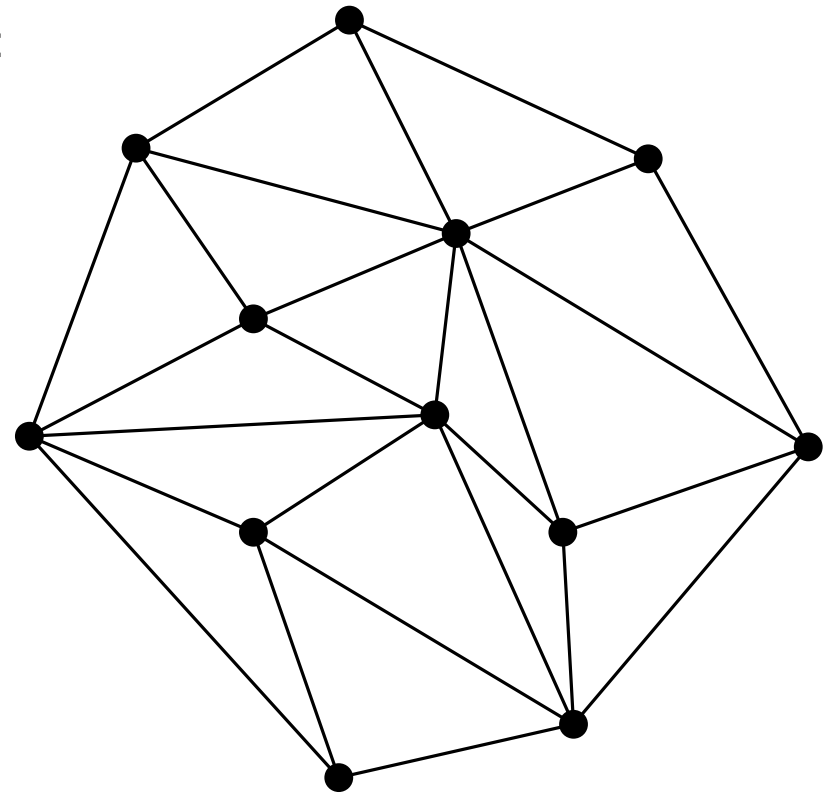
TRIANGULATING POINT SETS

DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:

- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**:



TRIANGULATING POINT SETS

DATA STRUCTURE

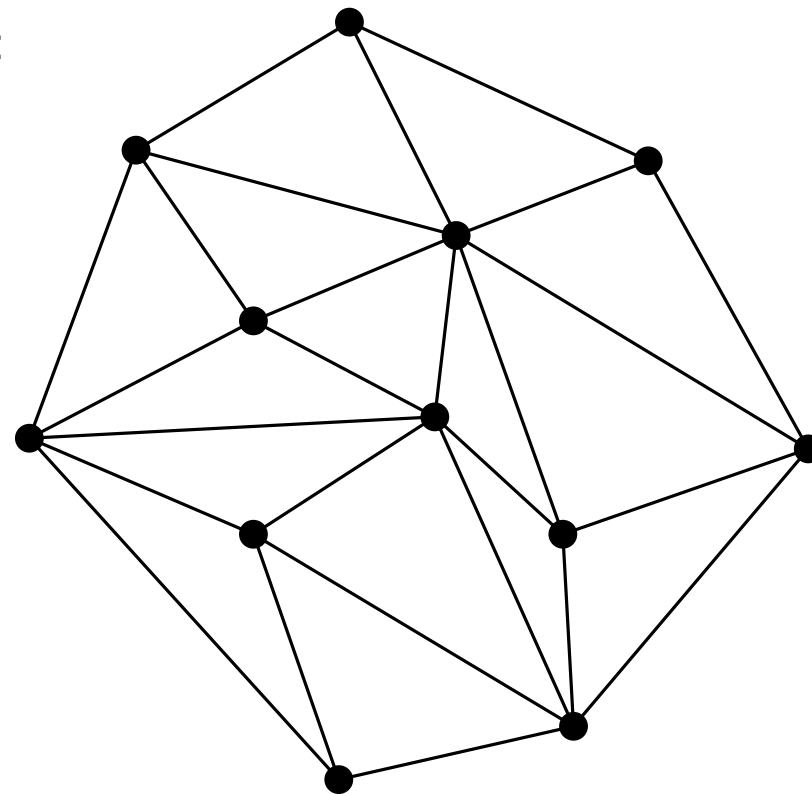
We want to answer the most usual questions for any decomposition of the plane:

- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**:

Table of vertices

v	x	y	e
-----	-----	-----	-----



TRIANGULATING POINT SETS

DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:

- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

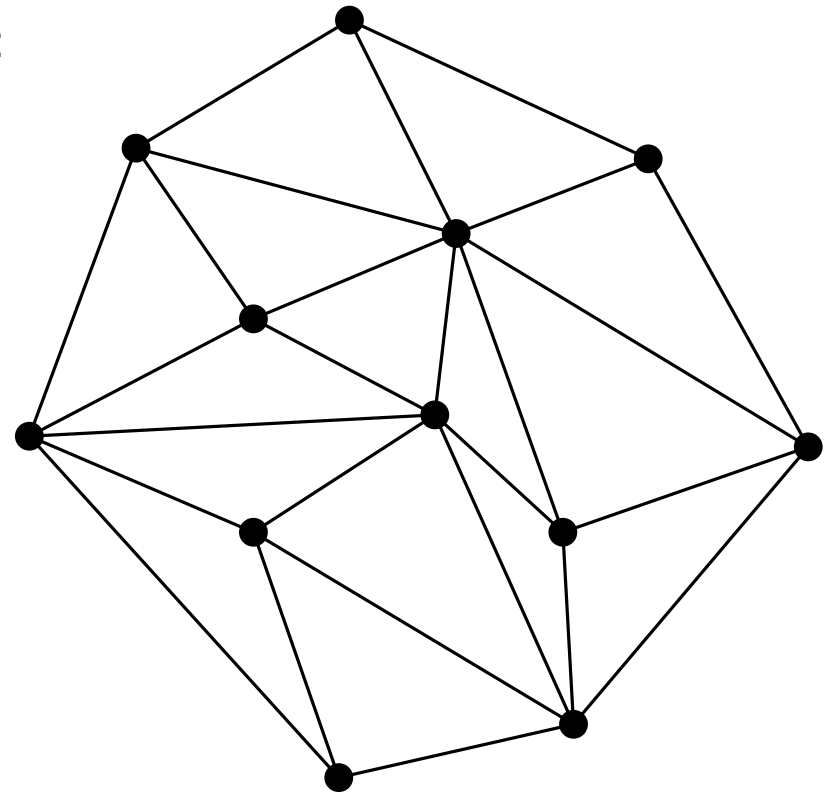
The appropriate structure is, once again, a **DCEL**:

Table of vertices

v	x	y	e
-----	-----	-----	-----

Table of faces

t	e
-----	-----



TRIANGULATING POINT SETS

DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:

- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**:

Table of vertices

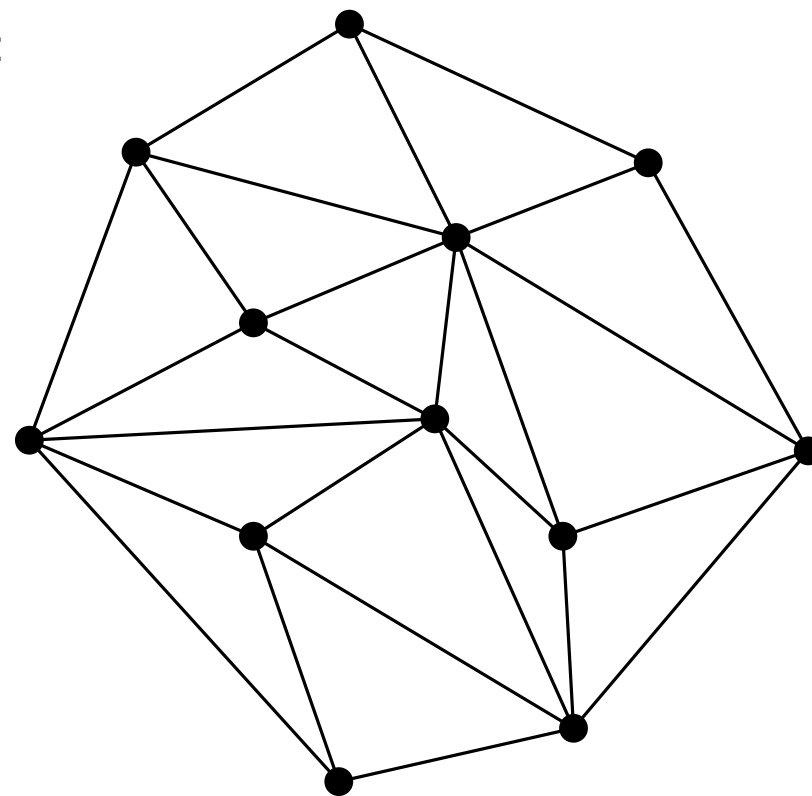
v	x	y	e
-----	-----	-----	-----

Table of faces

t	e
-----	-----

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
-----	-------	-------	-------	-------	-------	-------



TRIANGULATING POINT SETS

ALGORITHMS

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.

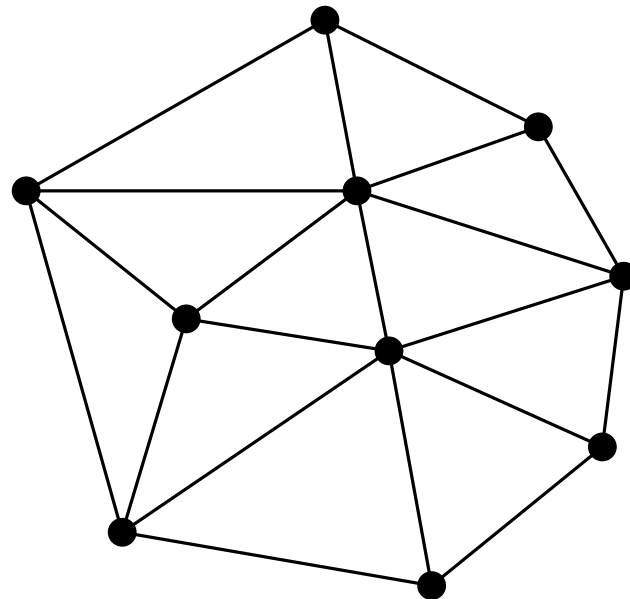
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



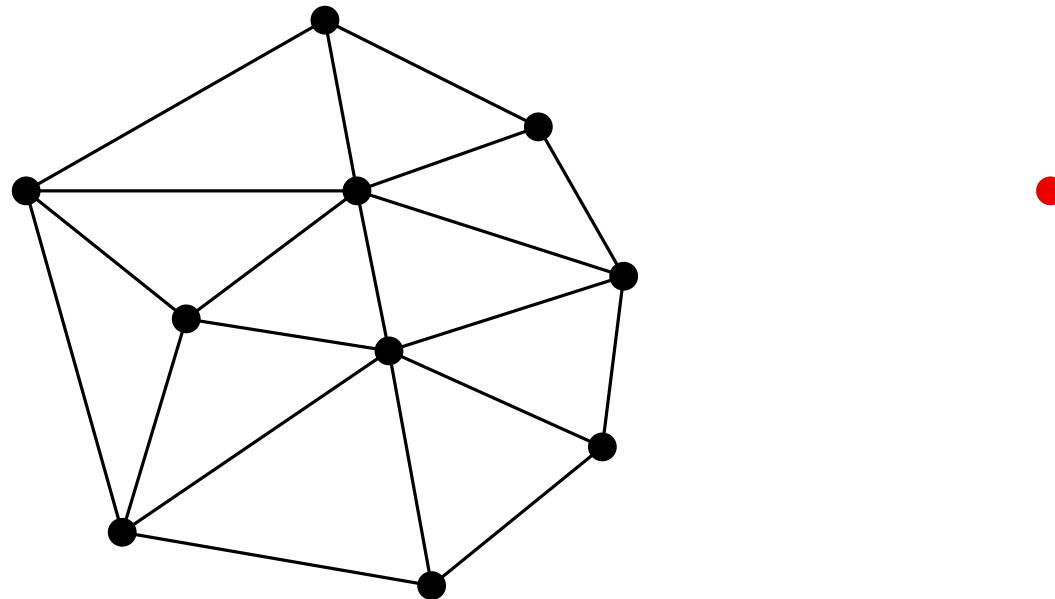
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



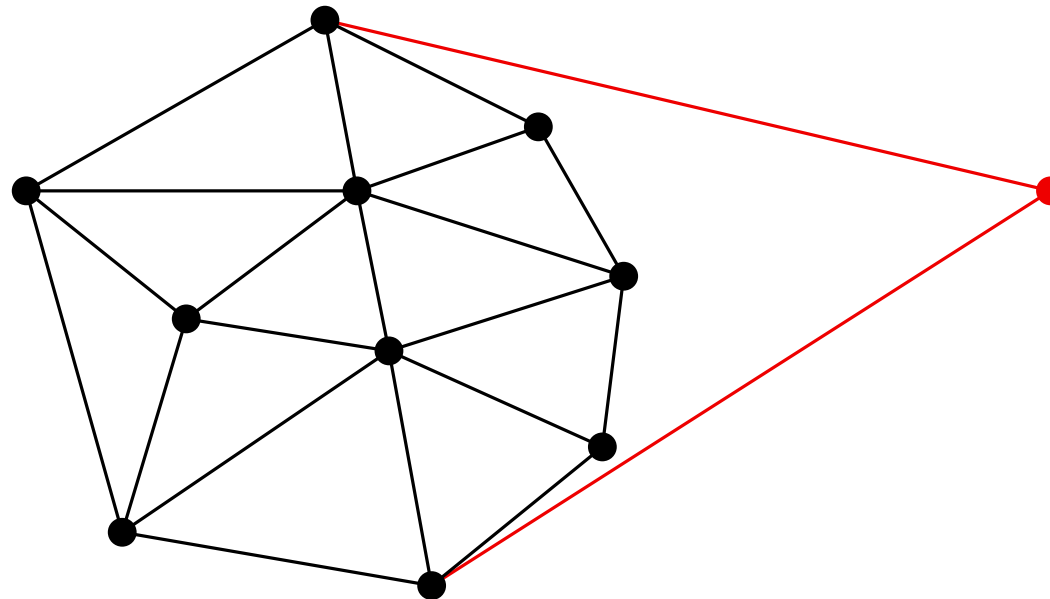
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



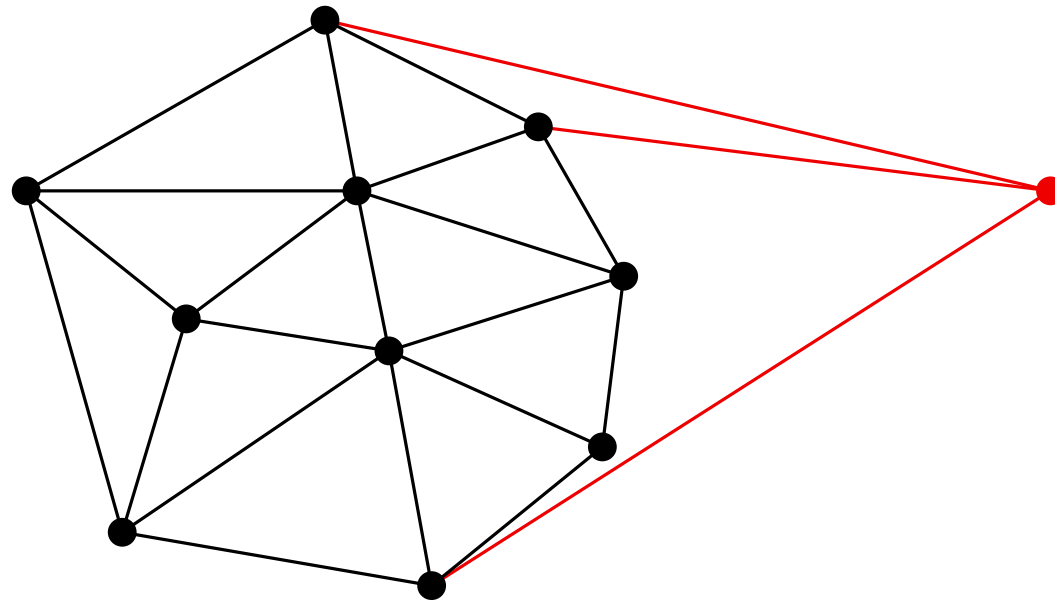
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



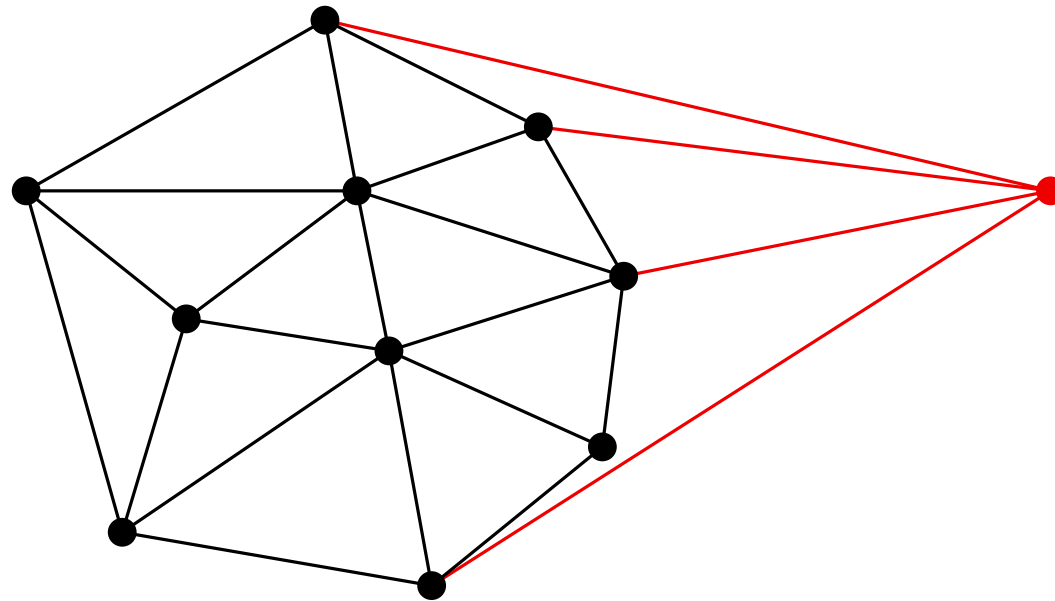
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



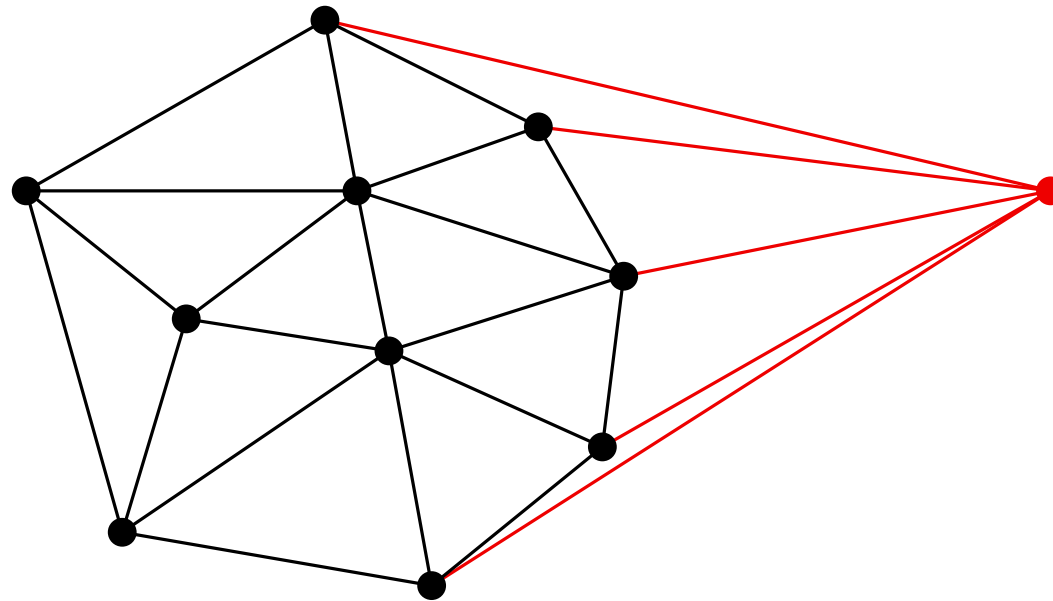
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



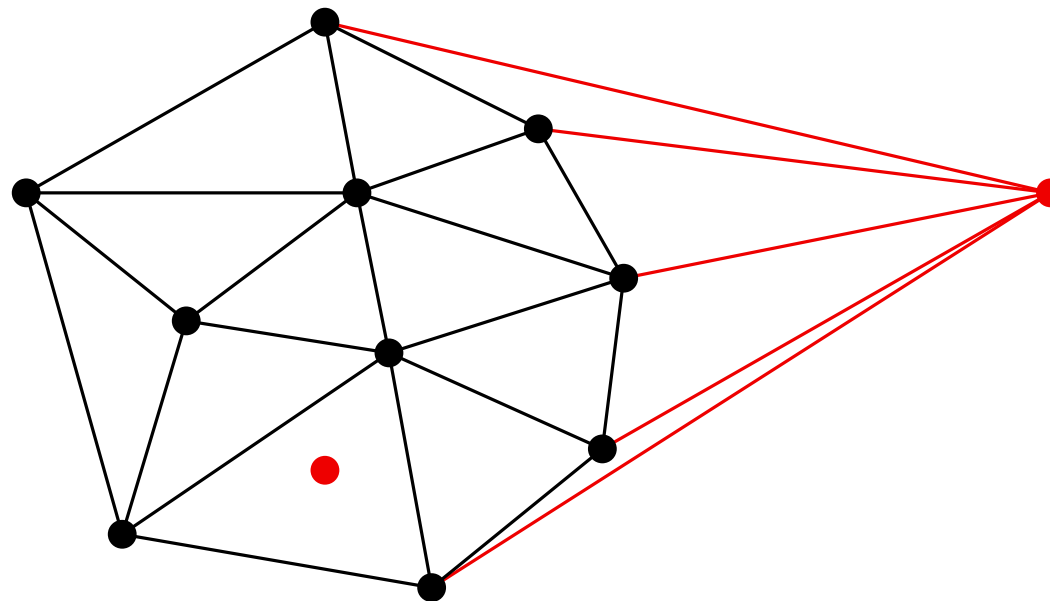
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



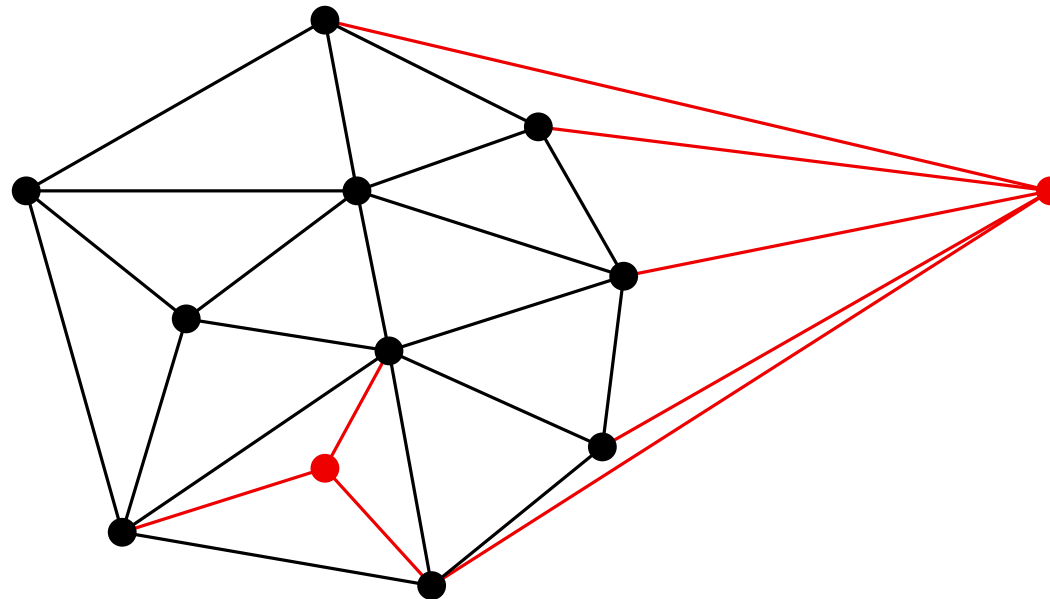
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



TRIANGULATING POINT SETS

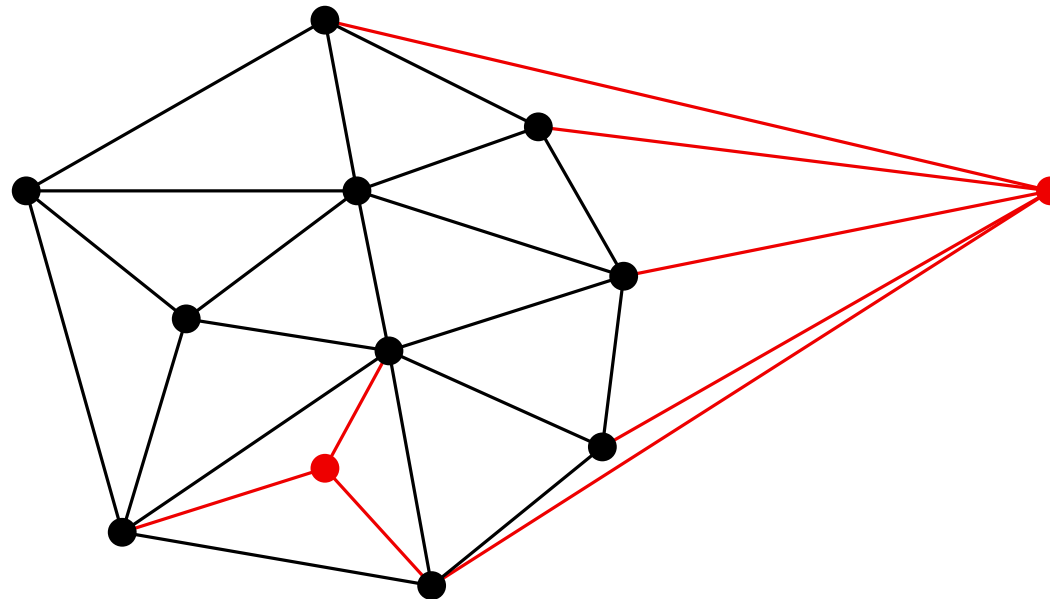
ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

For each i , detect whether p_i lies in the interior or the exterior of $ch(p_1, \dots, p_{i-1})$. If it is external, compute the supporting lines from p_i to $ch(p_1, \dots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle T containing p_i and partition T into 3 triangles.



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Start by sorting the points in lexicographical order in $O(n \log n)$ time. The information of the sorted order of the points allows to add the i diagonals in $O(i)$ time, so that the amortized cost of the insertion of all diagonals is done in $O(n)$ time.

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

Start by sorting the points in lexicographical order in $O(n \log n)$ time. The information of the sorted order of the points allows to add the i diagonals in $O(i)$ time, so that the amortized cost of the insertion of all diagonals is done in $O(n)$ time.

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

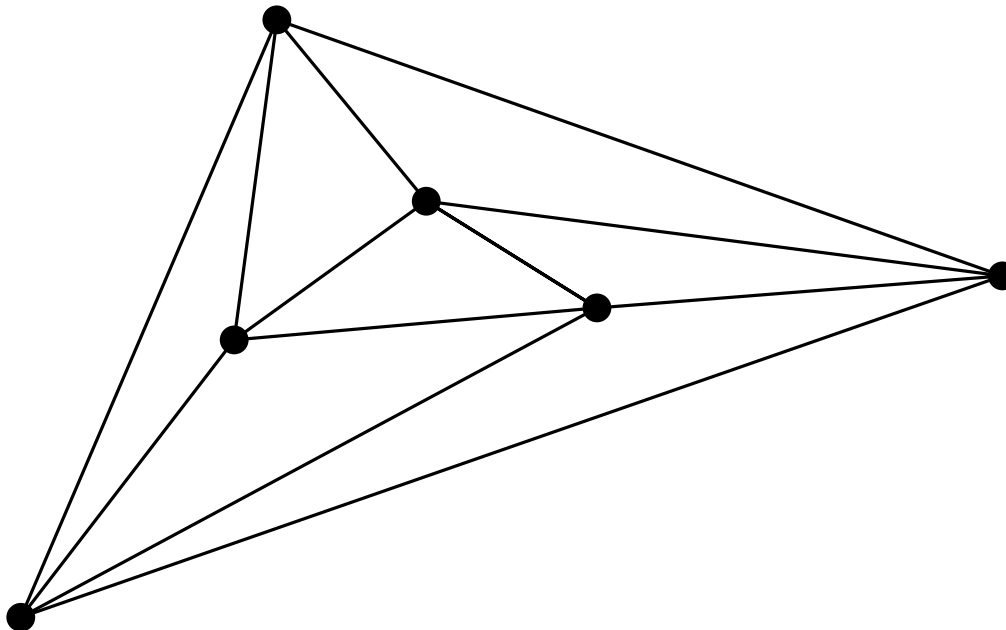
Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

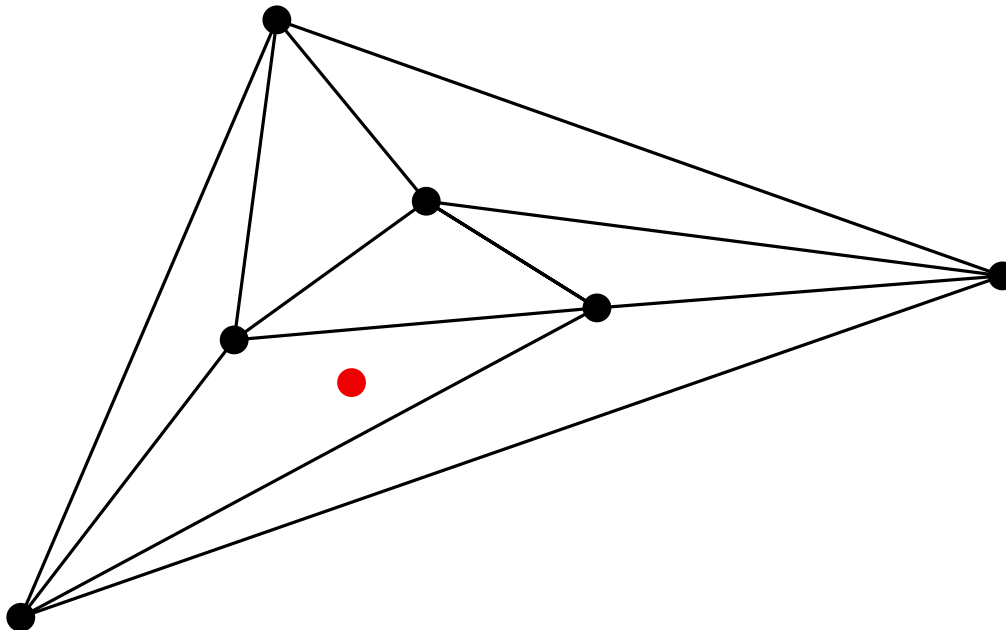
Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

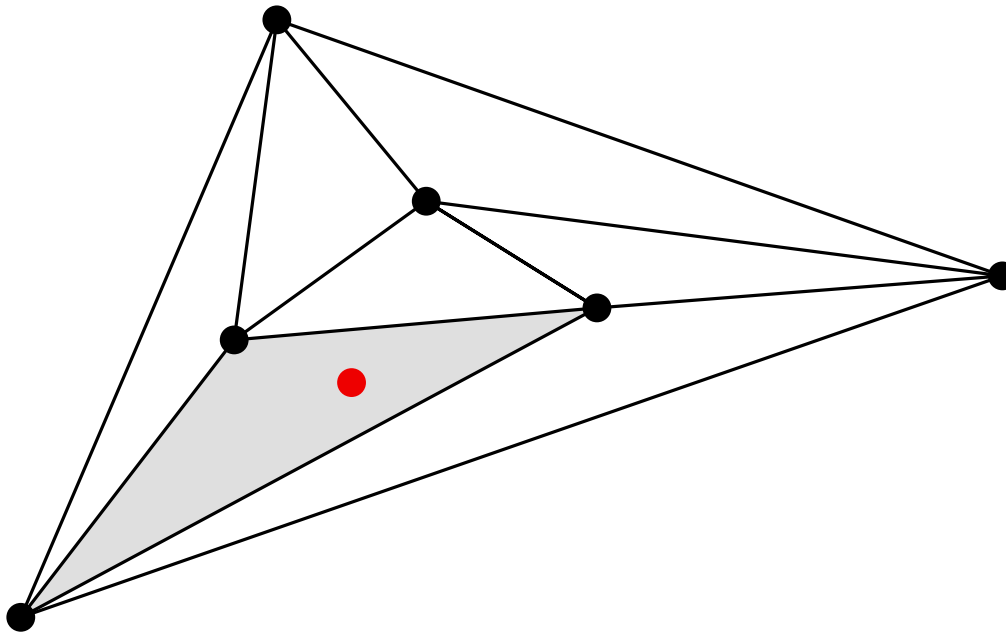
Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.



T

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

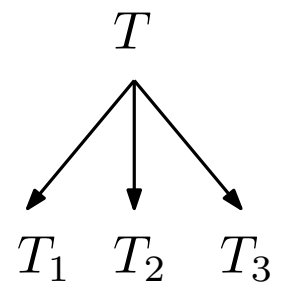
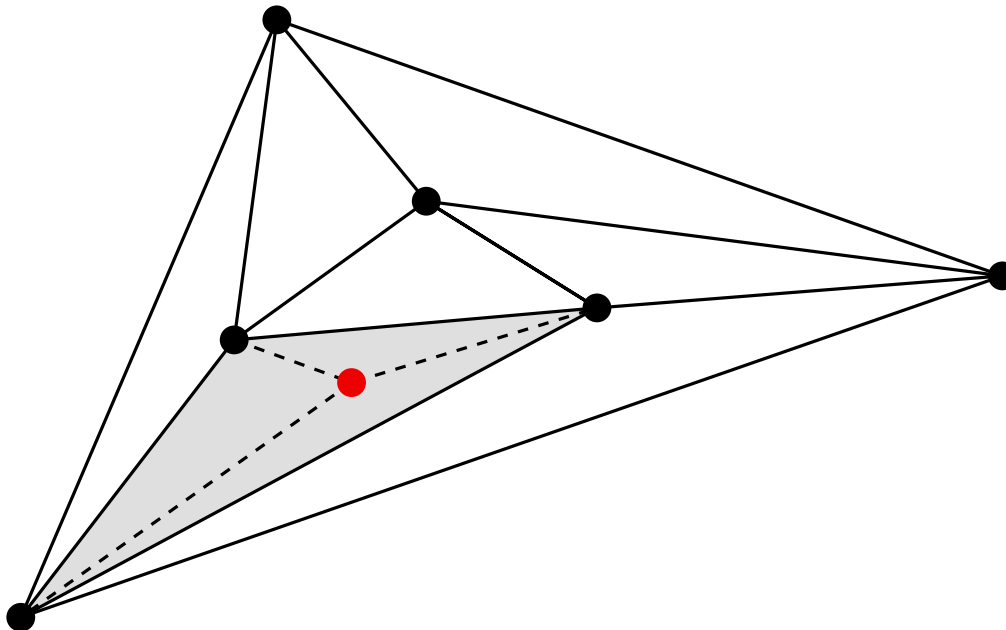
Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

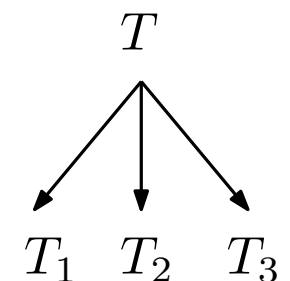
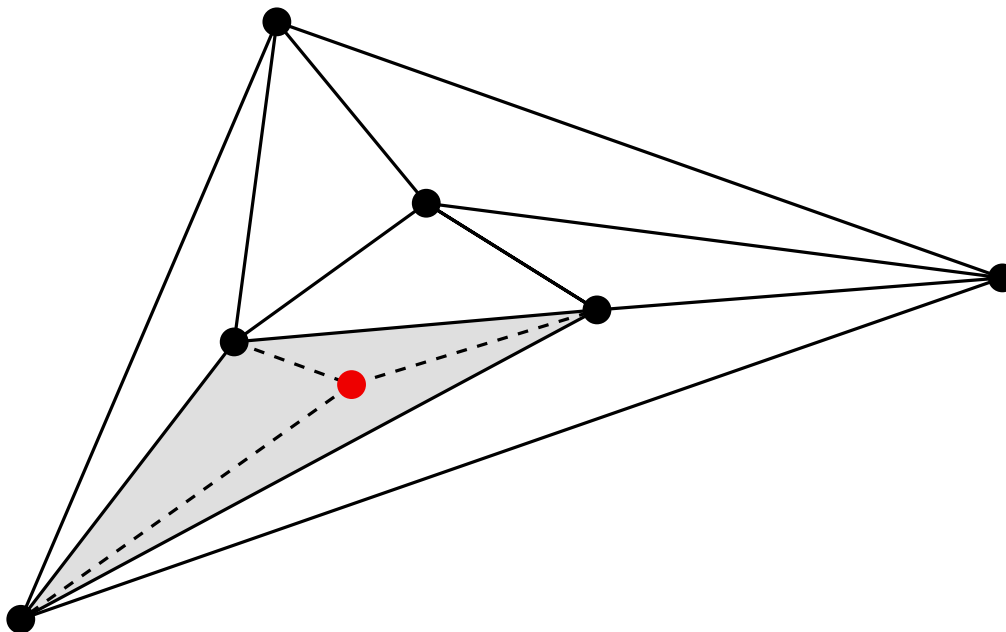
1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

- | | |
|----------------------------------|---|
| 1.1. Without sorting | Running time: $O(n^2)$ |
| 1.2. With sorting | Running time: $O(n \log n)$ |
| 1.3. With hierarchical structure | Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced |
| 1.4. Randomized | Running time: $O(n \log n)$ expected |
| 1.5. With auxiliary point(s) | |

A fixed point p is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

When inserting each point p_i :

- Scan the triangles stabbed by the segment $\overline{pp_i}$.
- Update, if necessary, the information of the triangle containing p .

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

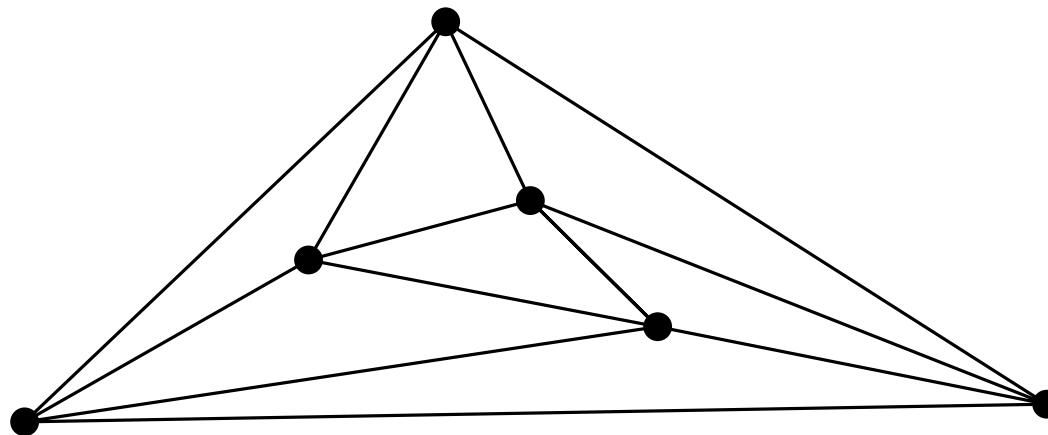
Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

A fixed point p is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

When inserting each point p_i :

- Scan the triangles stabbed by the segment $\overline{pp_i}$.
- Update, if necessary, the information of the triangle containing p .



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

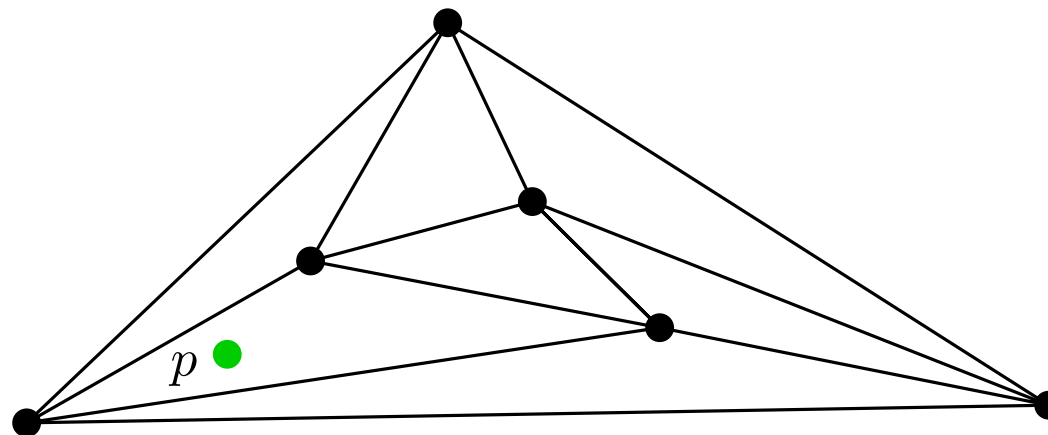
Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

A fixed point p is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

When inserting each point p_i :

- Scan the triangles stabbed by the segment $\overline{pp_i}$.
- Update, if necessary, the information of the triangle containing p .



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

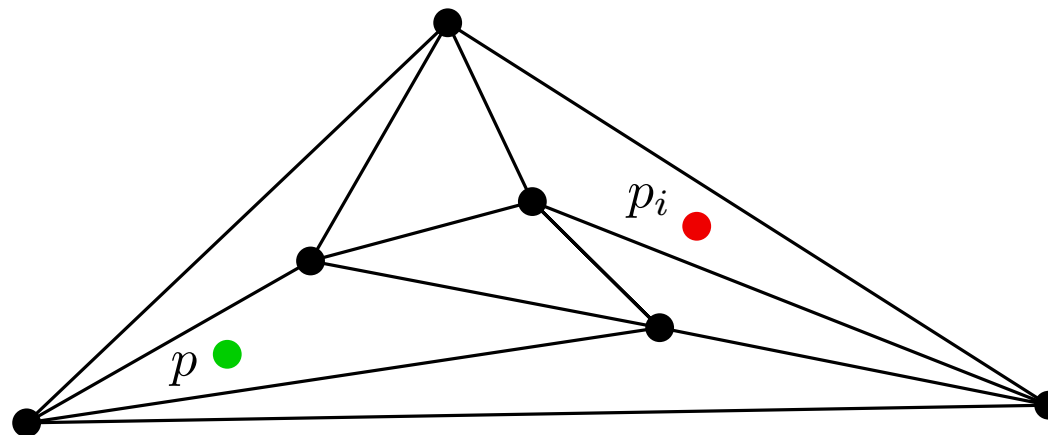
Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

A fixed point p is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

When inserting each point p_i :

- Scan the triangles stabbed by the segment $\overline{pp_i}$.
- Update, if necessary, the information of the triangle containing p .



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

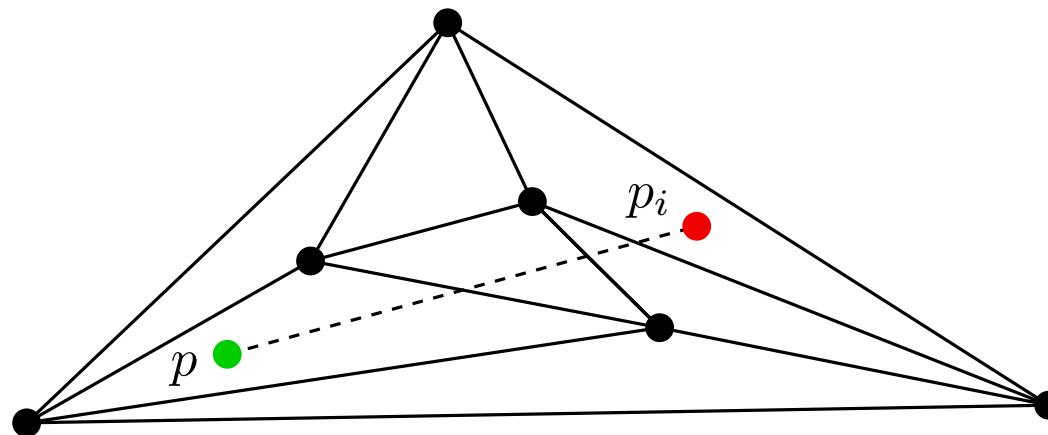
Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

A fixed point p is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

When inserting each point p_i :

- Scan the triangles stabbed by the segment $\overline{pp_i}$.
- Update, if necessary, the information of the triangle containing p .



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

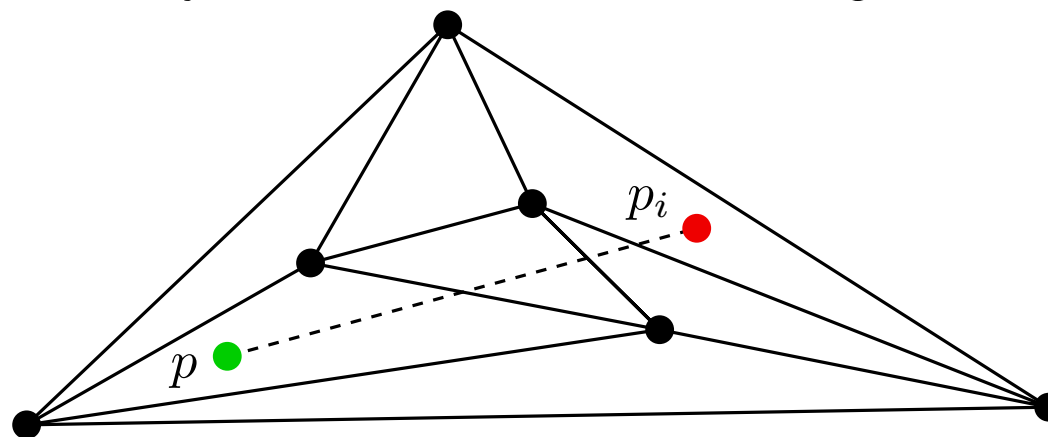
1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

A fixed point p is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

When inserting each point p_i :

- Scan the triangles stabbed by the segment $\overline{pp_i}$.
- Update, if necessary, the information of the triangle containing p .



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

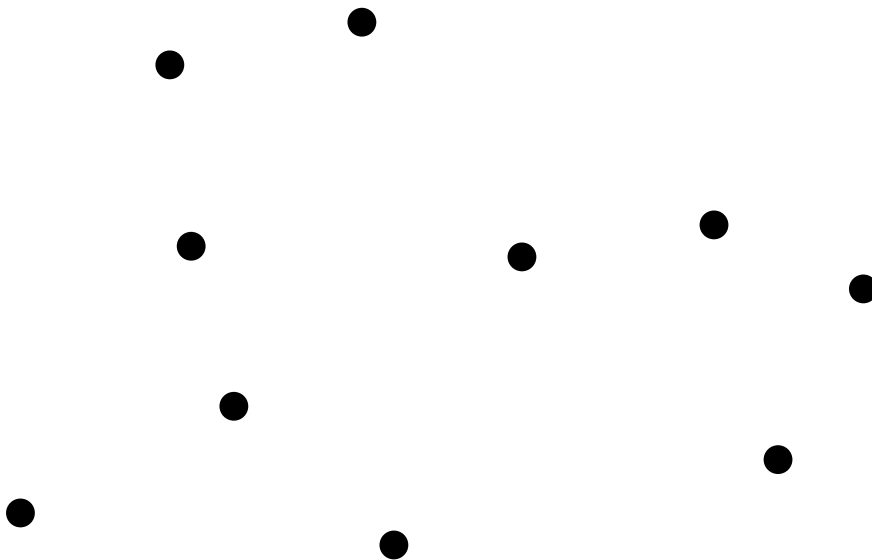
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

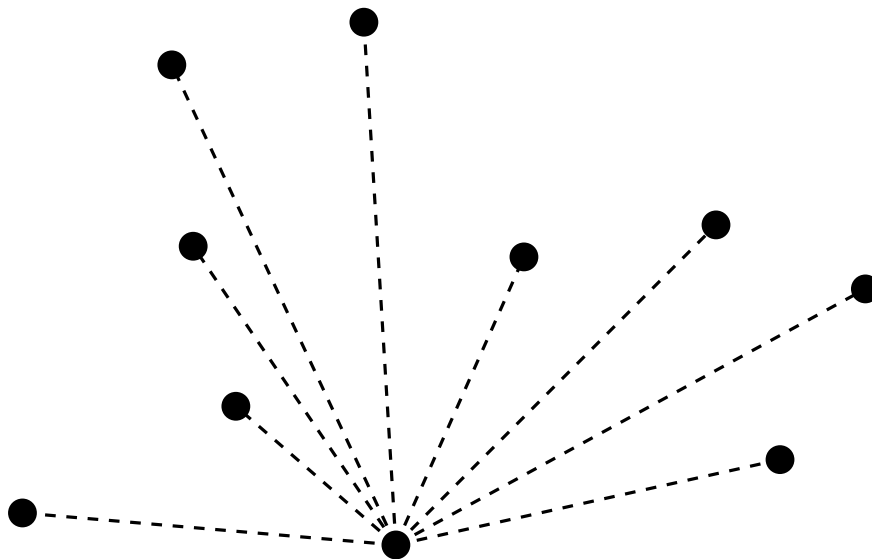
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

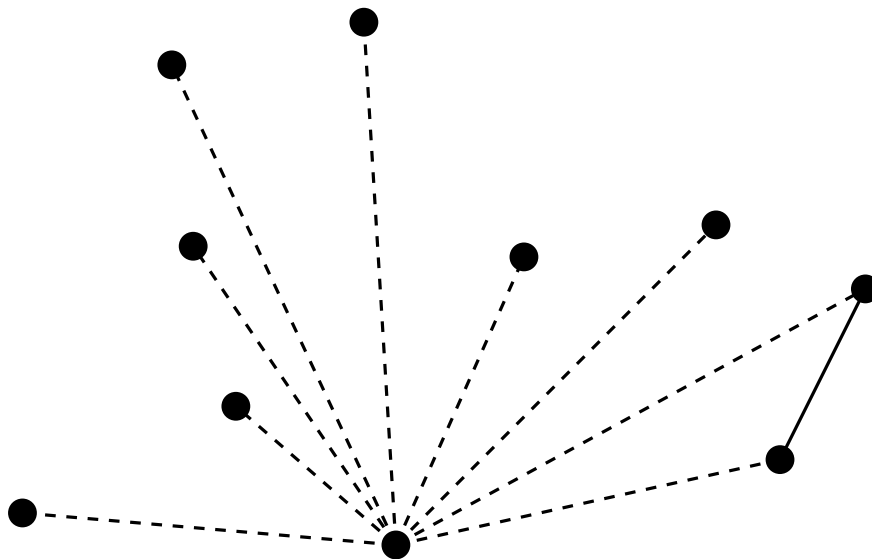
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

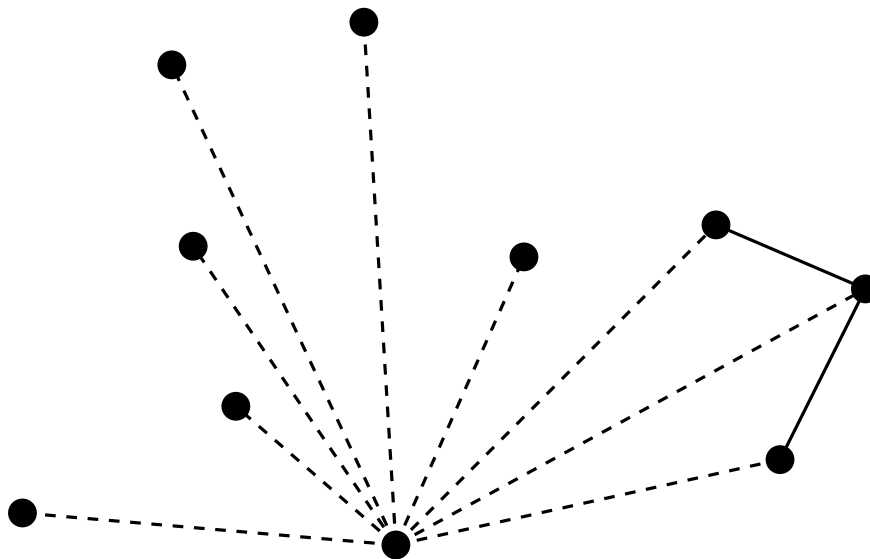
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

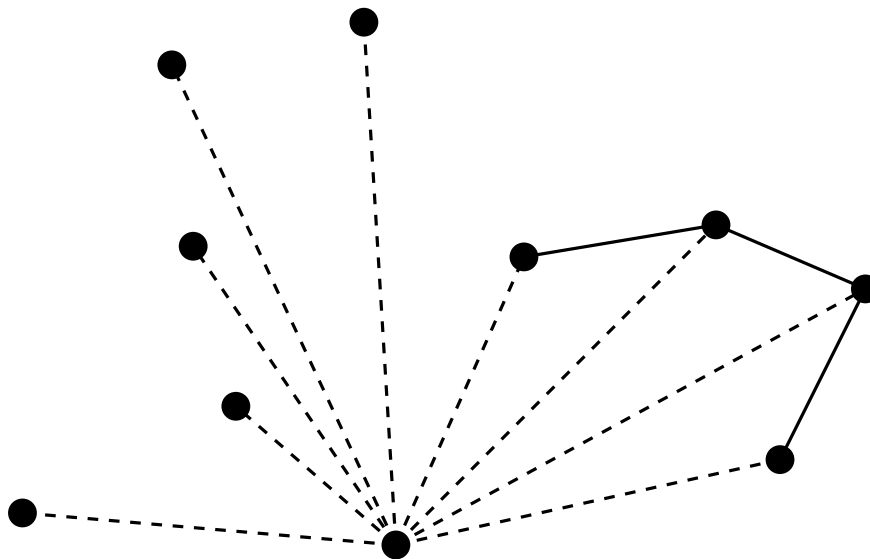
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

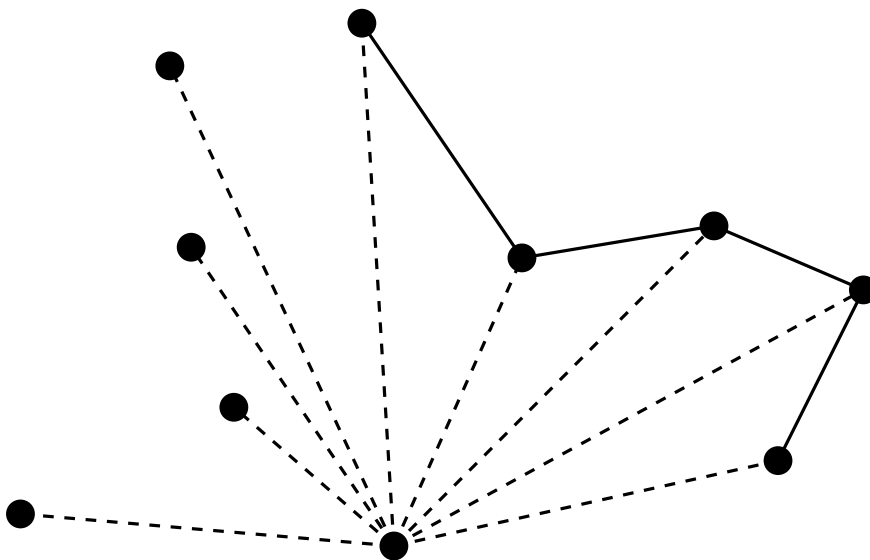
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

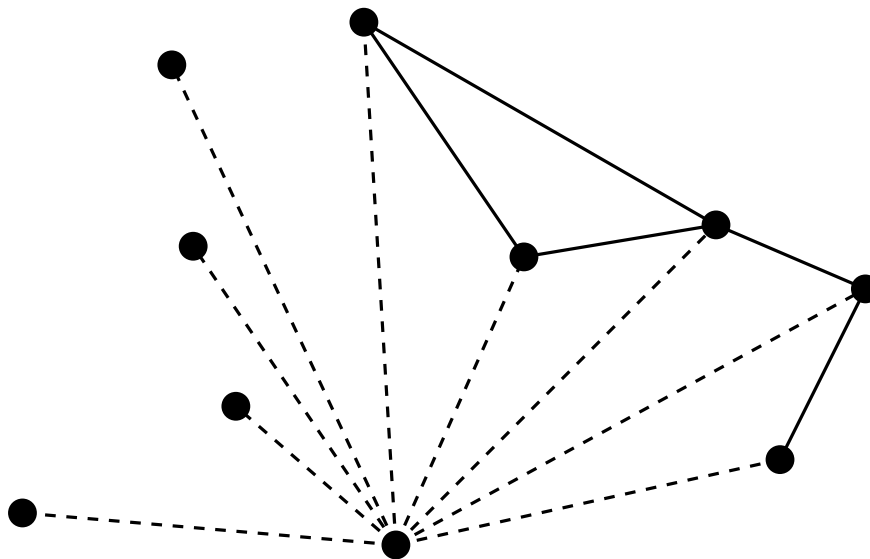
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

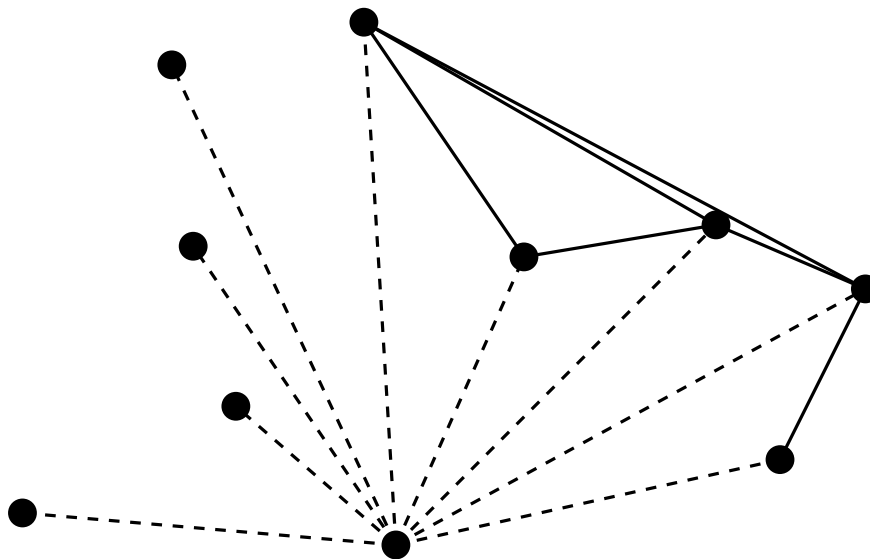
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

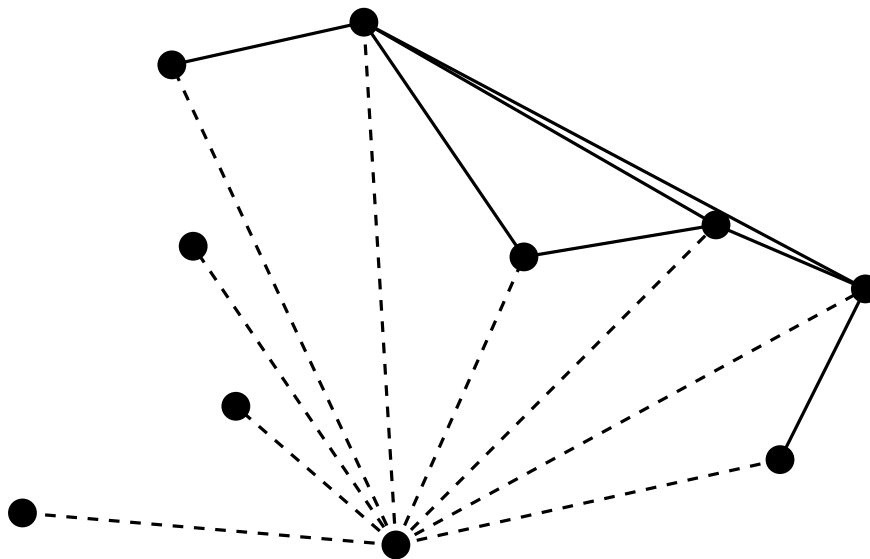
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

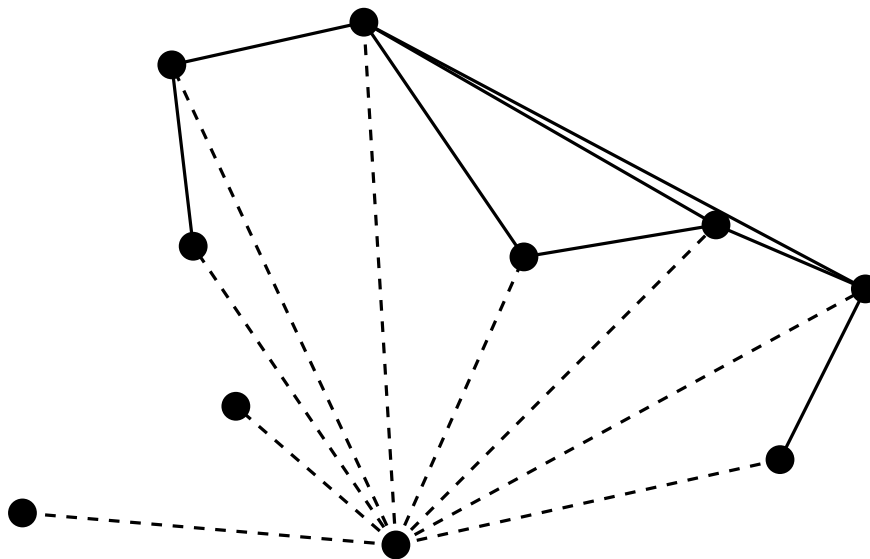
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

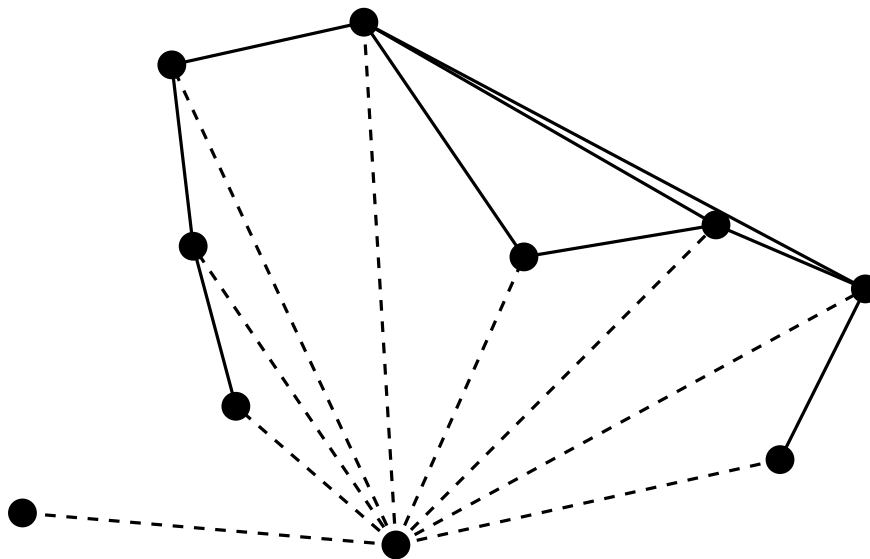
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

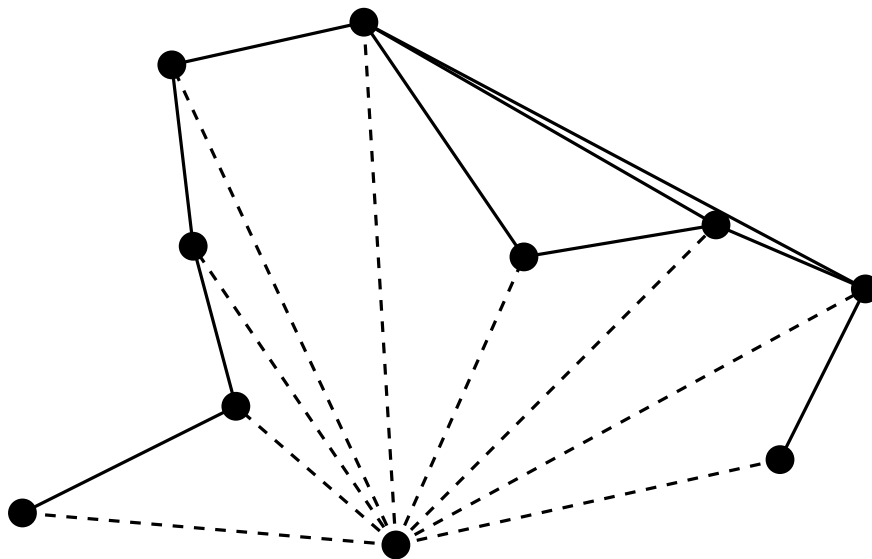
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

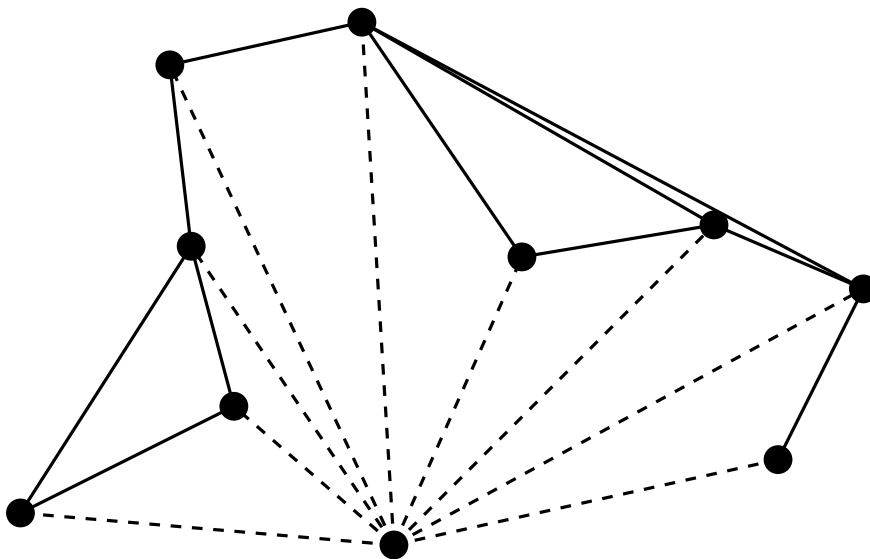
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

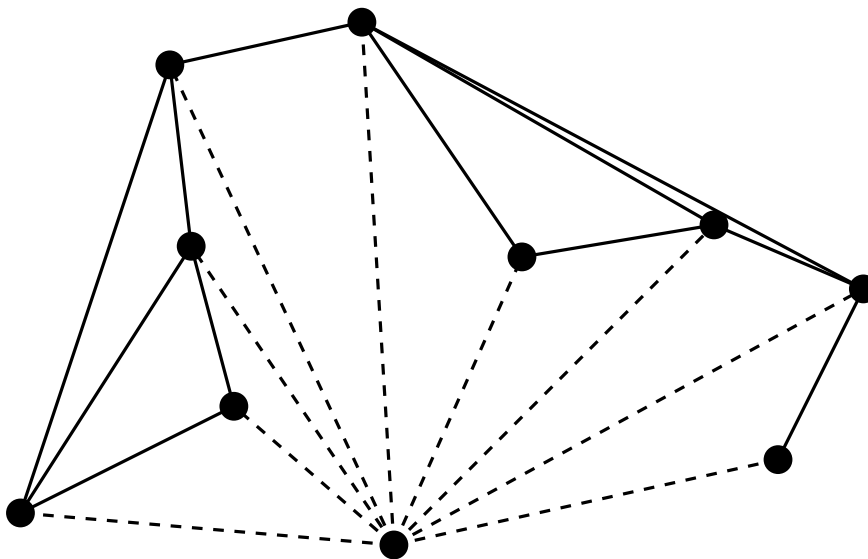
1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

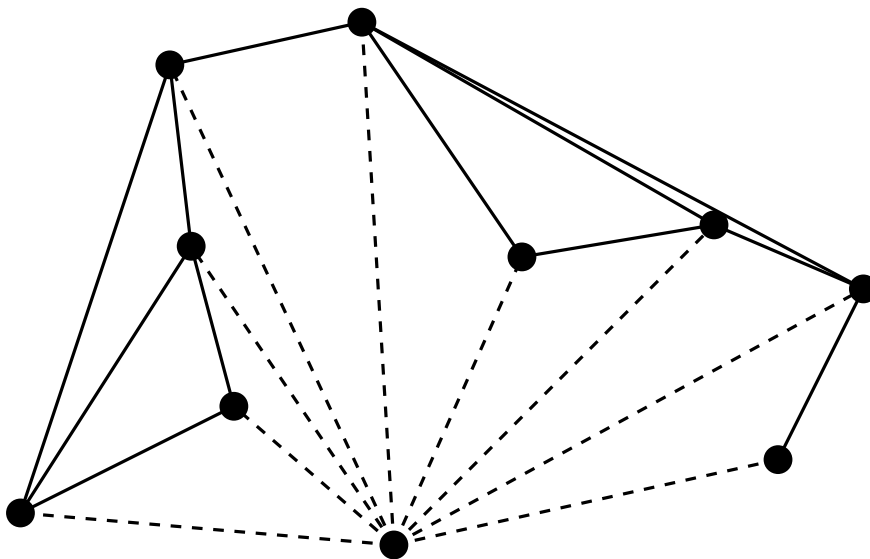
Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm

Running time: $O(n \log n)$



TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm

Running time: $O(n \log n)$

3. Divide and conquer

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm

Running time: $O(n \log n)$

3. Divide and conquer

Initialization

Sort the points by abscissa

Advance

- Partition: divide the points into roughly two vertically separated halves
- Recursion: recursively triangulate each half
- Fusion: compute the external common tangents and triangulate the intermediate space

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm

Running time: $O(n \log n)$

3. Divide and conquer

Running time: $O(n \log n)$

Initialization

Sort the points by abscissa

Advance

- Partition: divide the points into roughly two vertically separated halves
- Recursion: recursively triangulate each half
- Fusion: compute the external common tangents and triangulate the intermediate space

TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

Running time: $O(n^2)$

1.2. With sorting

Running time: $O(n \log n)$

1.3. With hierarchical structure

Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced

1.4. Randomized

Running time: $O(n \log n)$ expected

1.5. With auxiliary point(s)

Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham's algorithm

Running time: $O(n \log n)$

3. Divide and conquer

Running time: $O(n \log n)$

LOWER BOUND

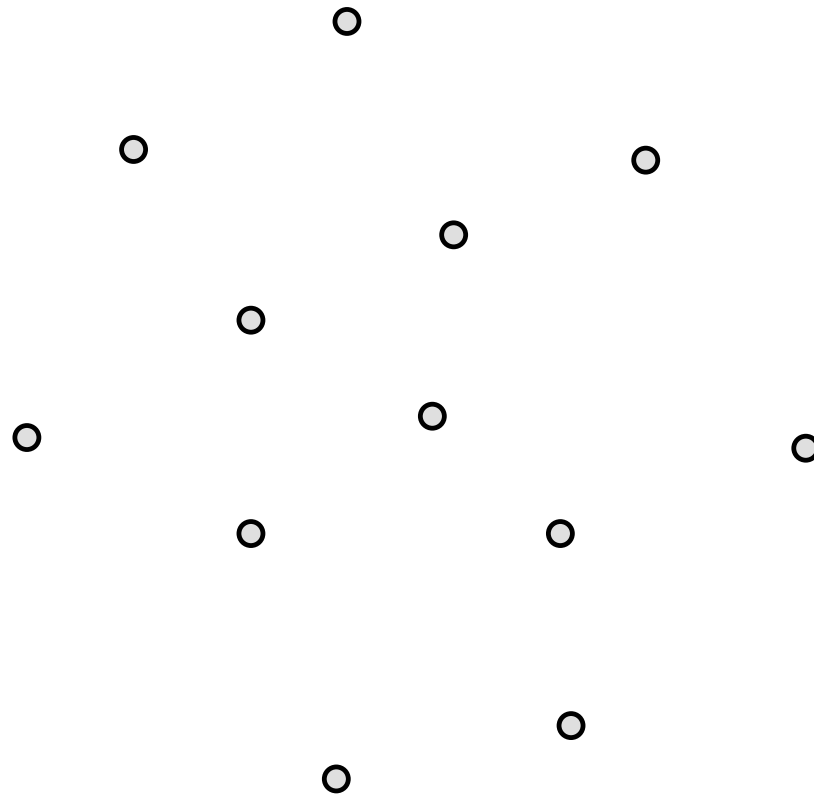
This problem has an $\Omega(n \log n)$ lower bound, since the convex hull of the set of points can be trivially obtained in $O(n)$ time from the triangulation.

TRIANGULATING POINT SETS

Quality of a triangulation

TRIANGULATING POINT SETS

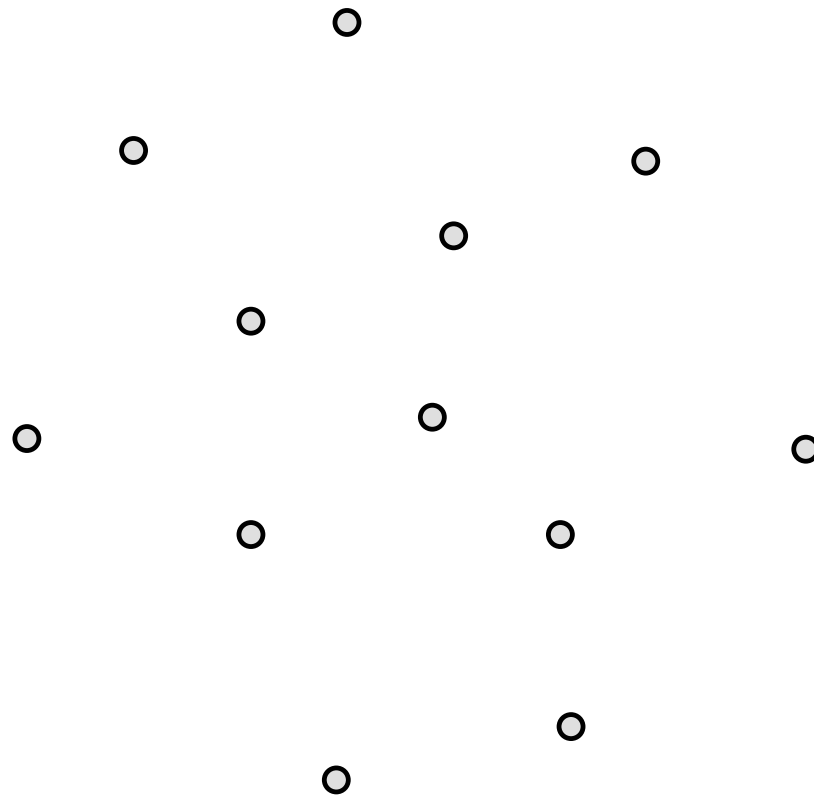
Quality of a triangulation



TRIANGULATING POINT SETS

Quality of a triangulation

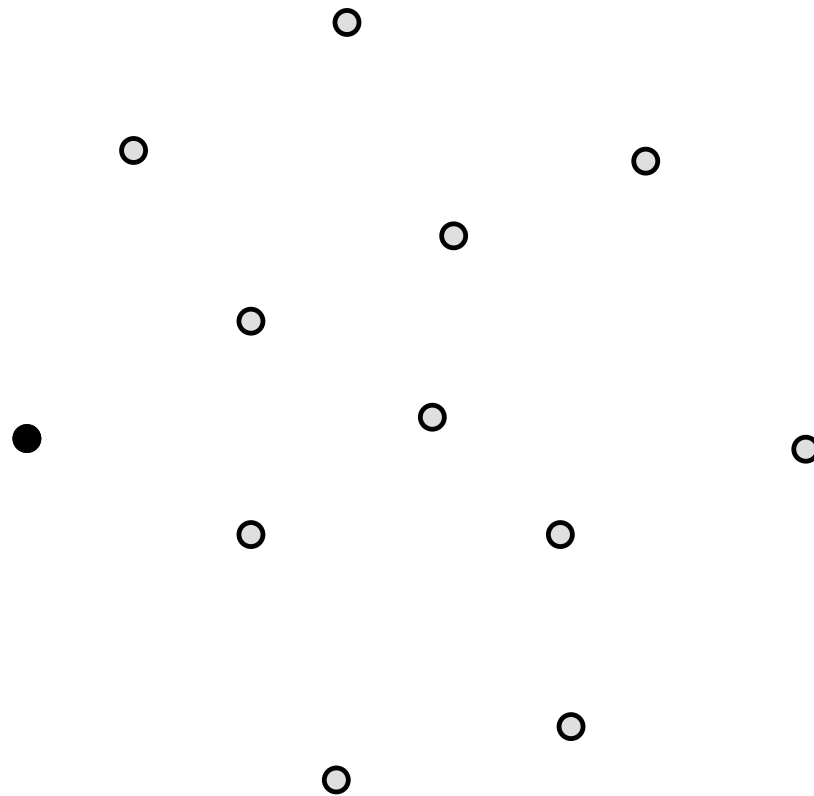
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

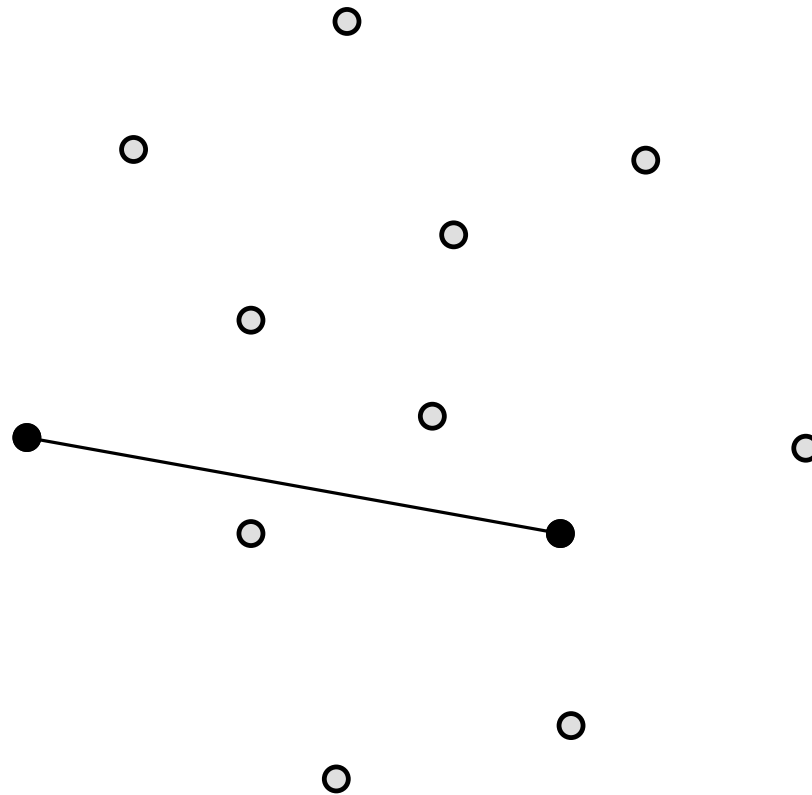
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

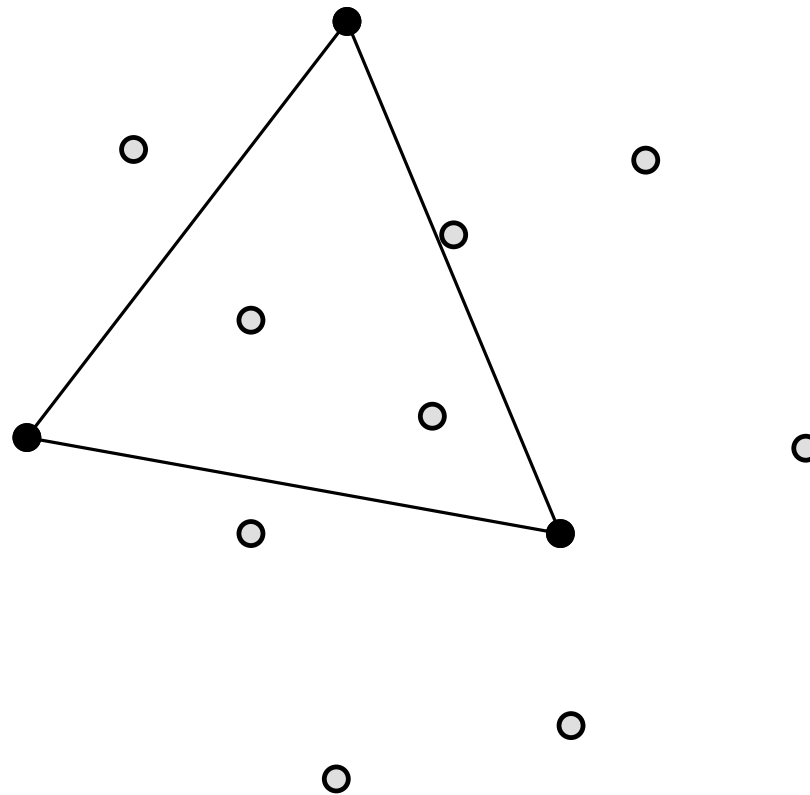
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

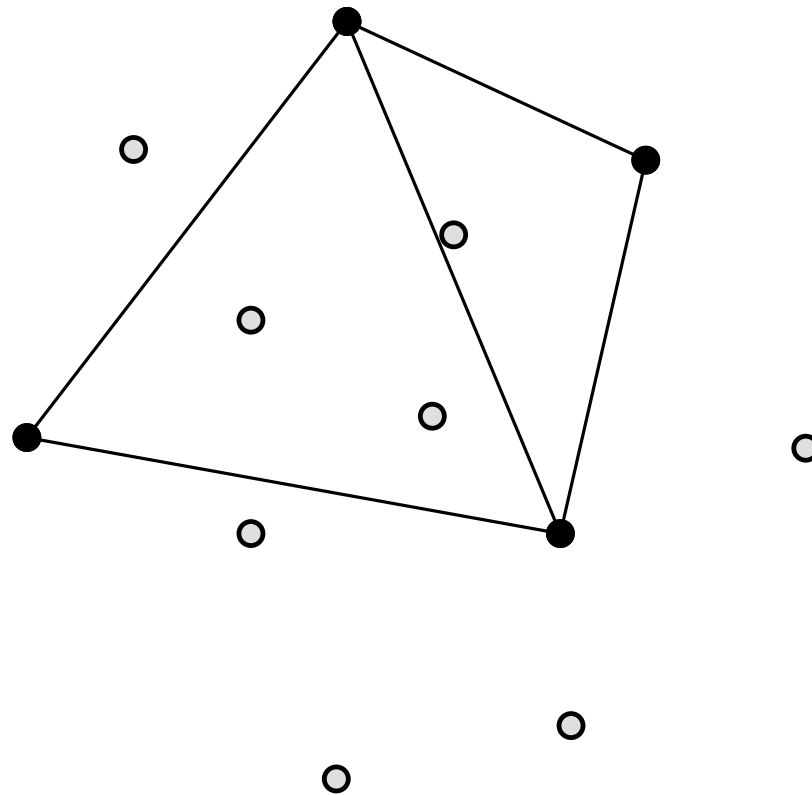
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

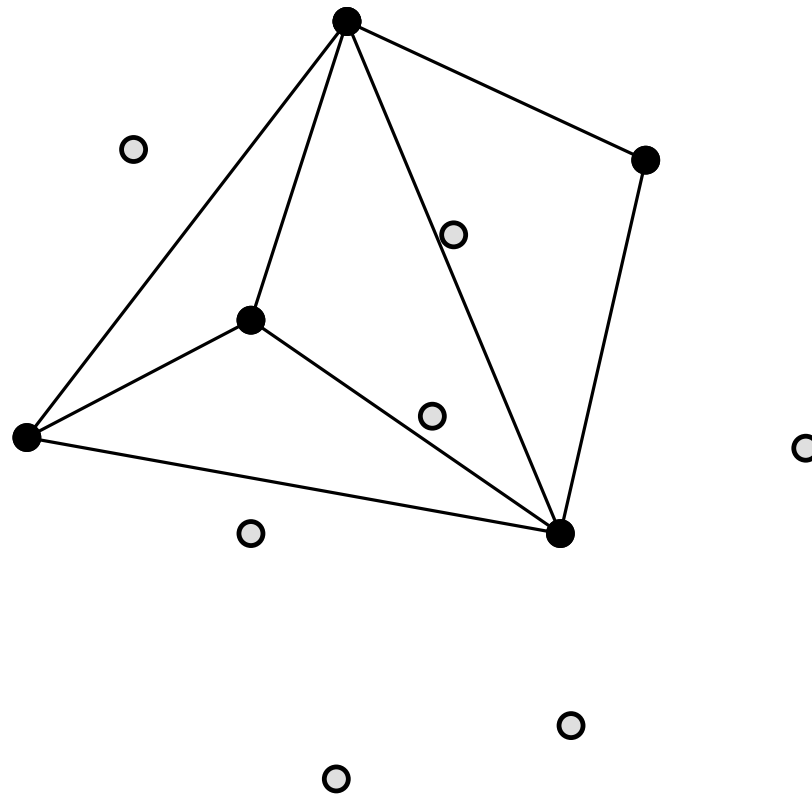
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

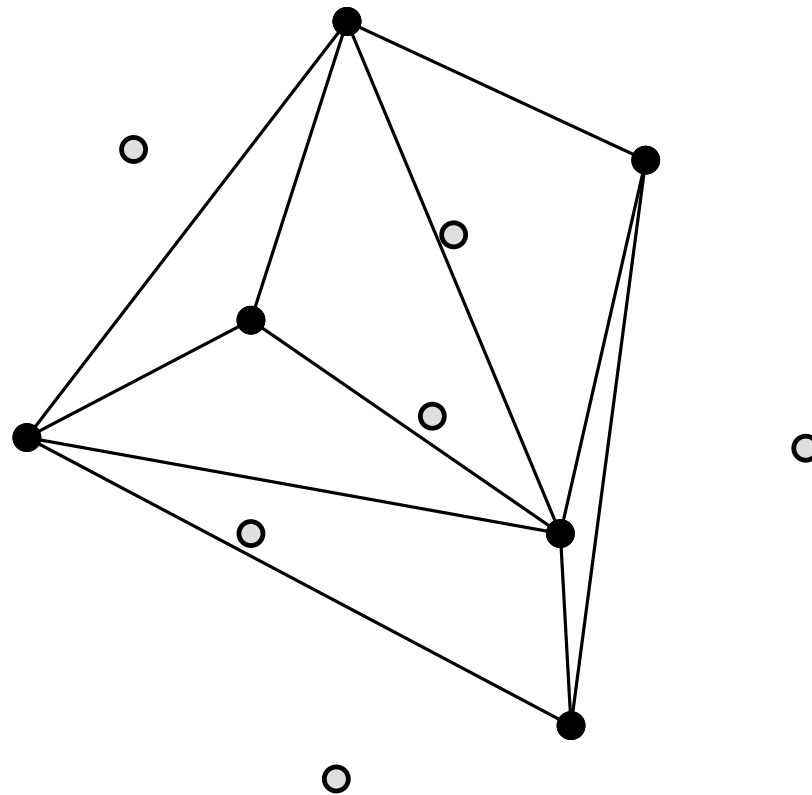
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

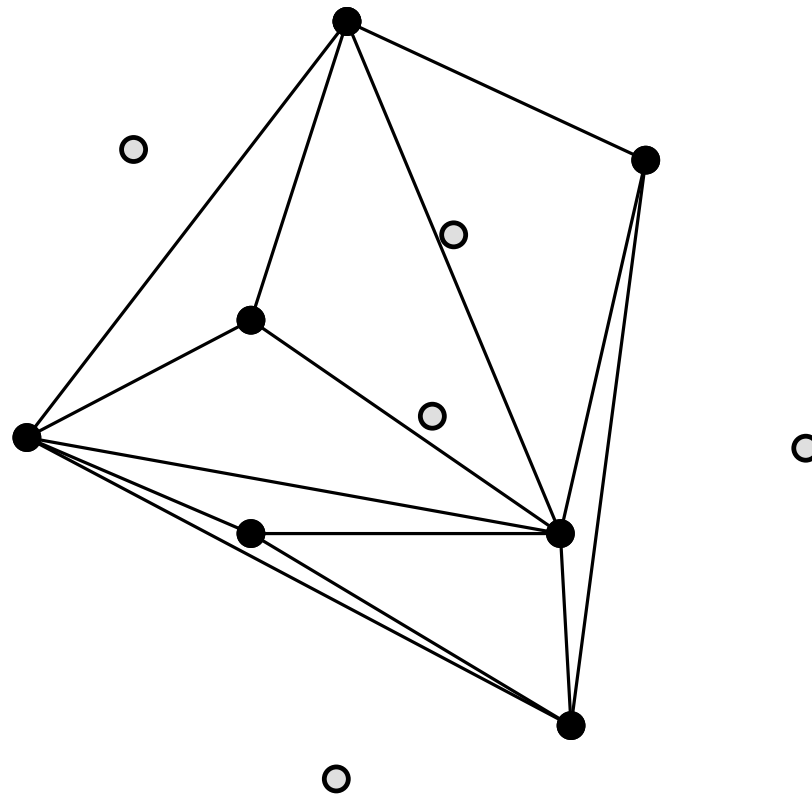
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

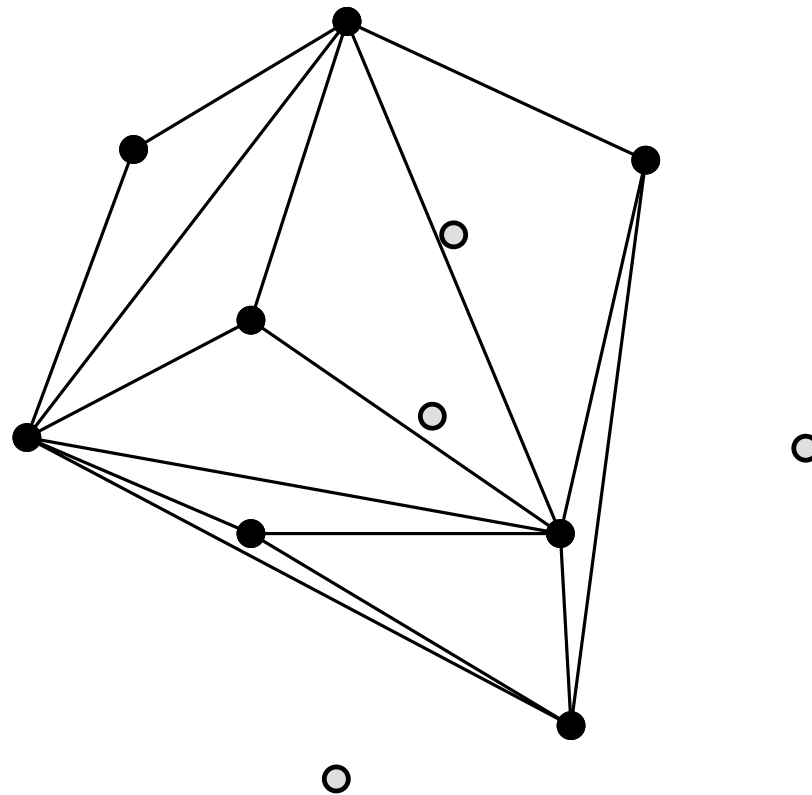
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

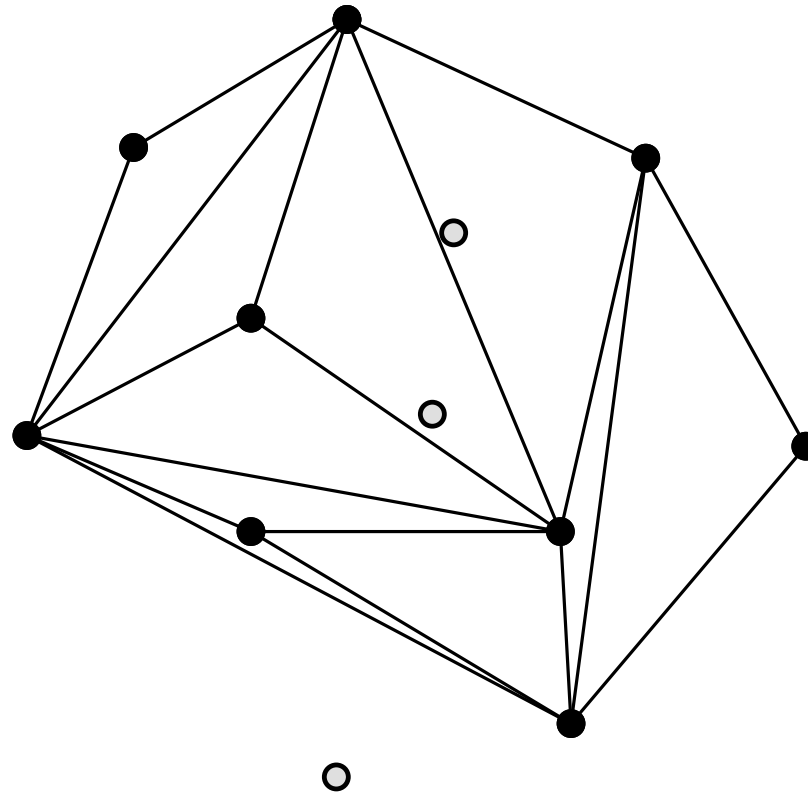
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

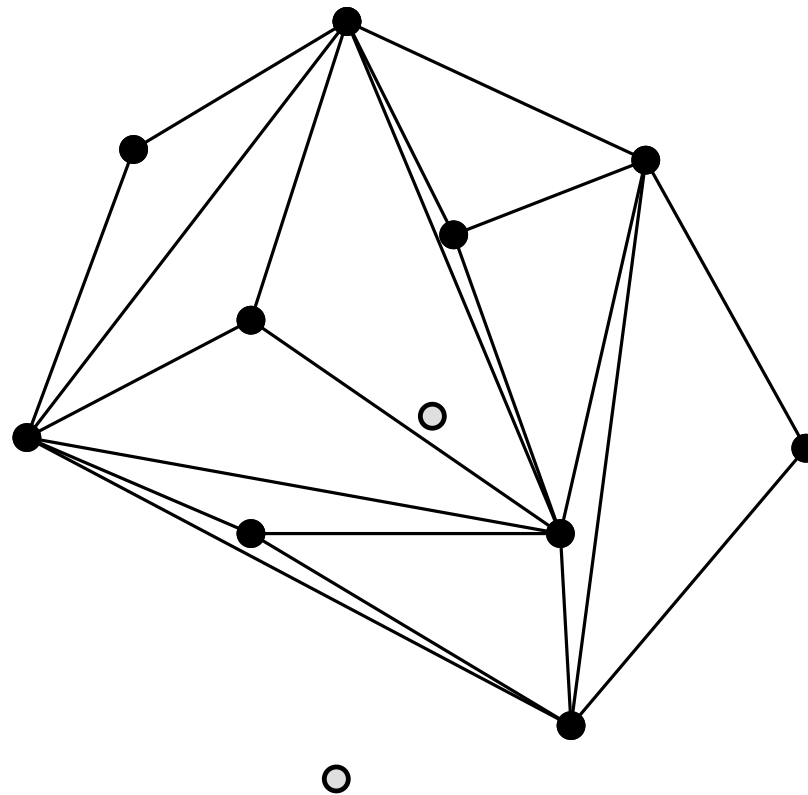
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

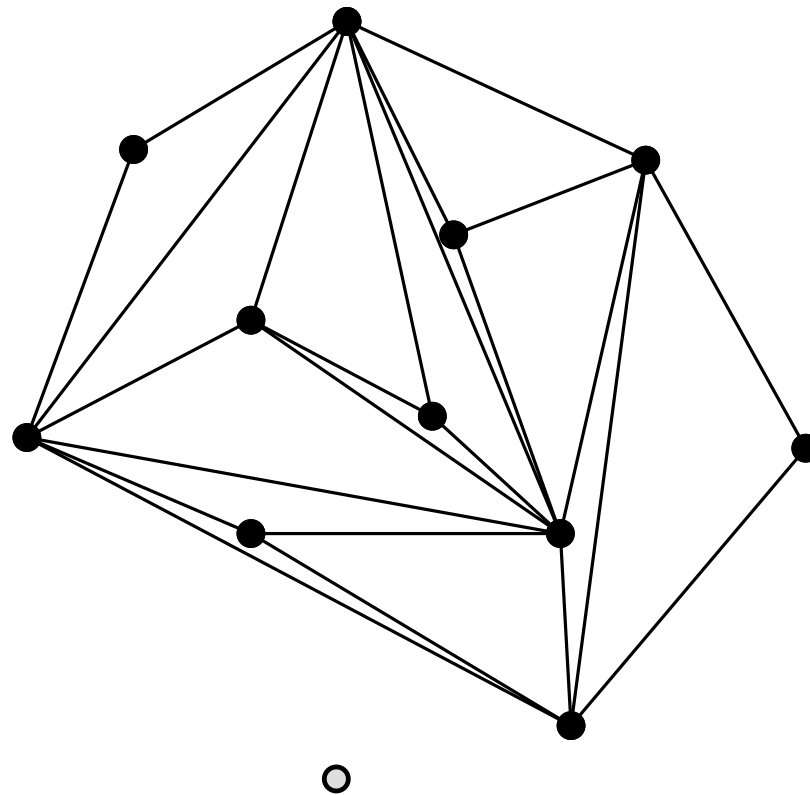
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

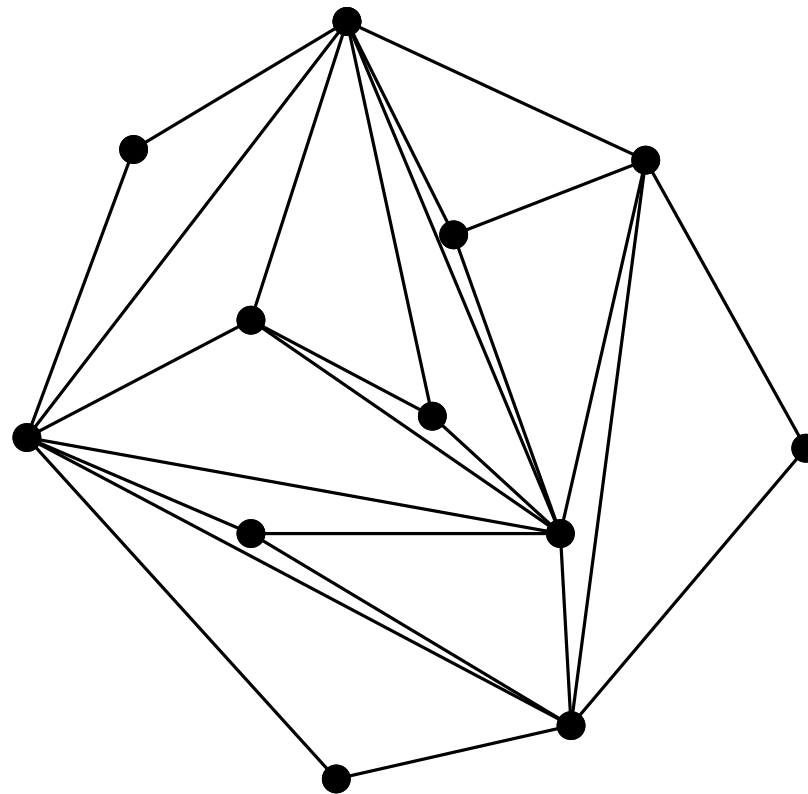
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

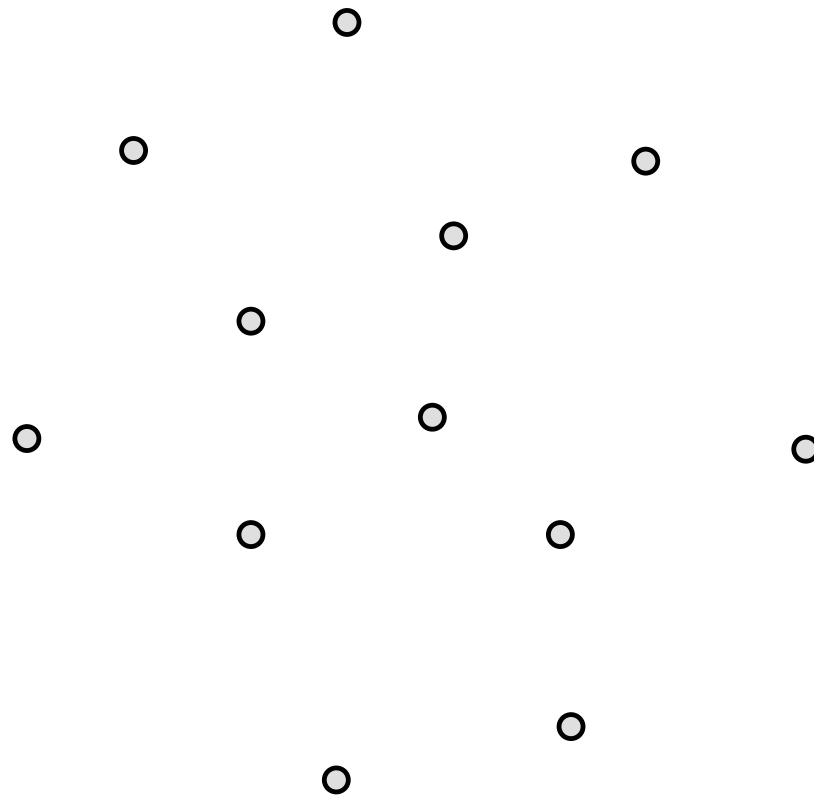
Incremental, without sorting



TRIANGULATING POINT SETS

Quality of a triangulation

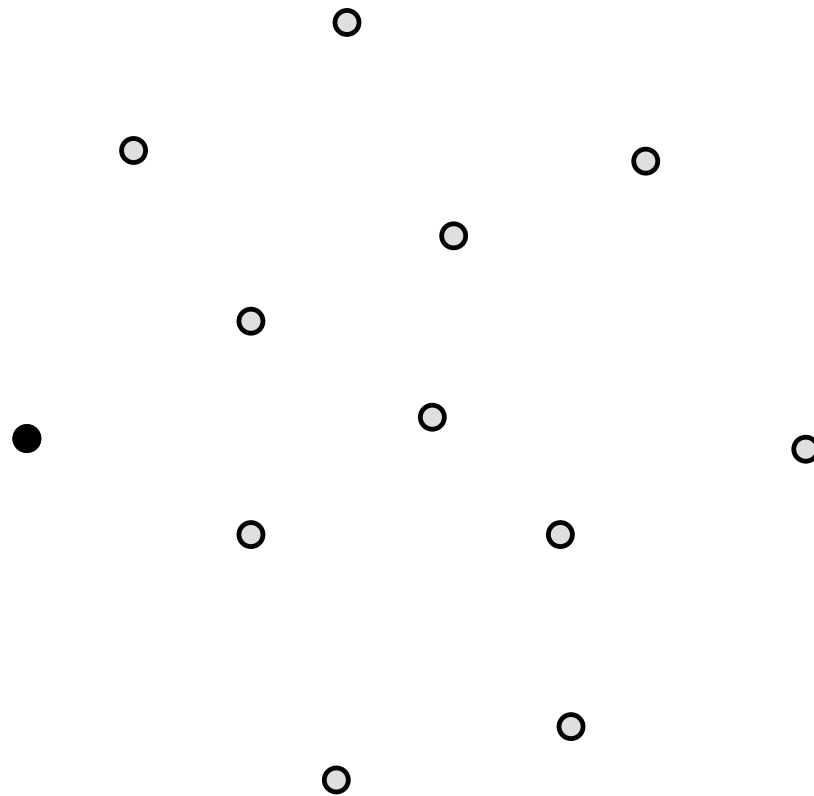
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

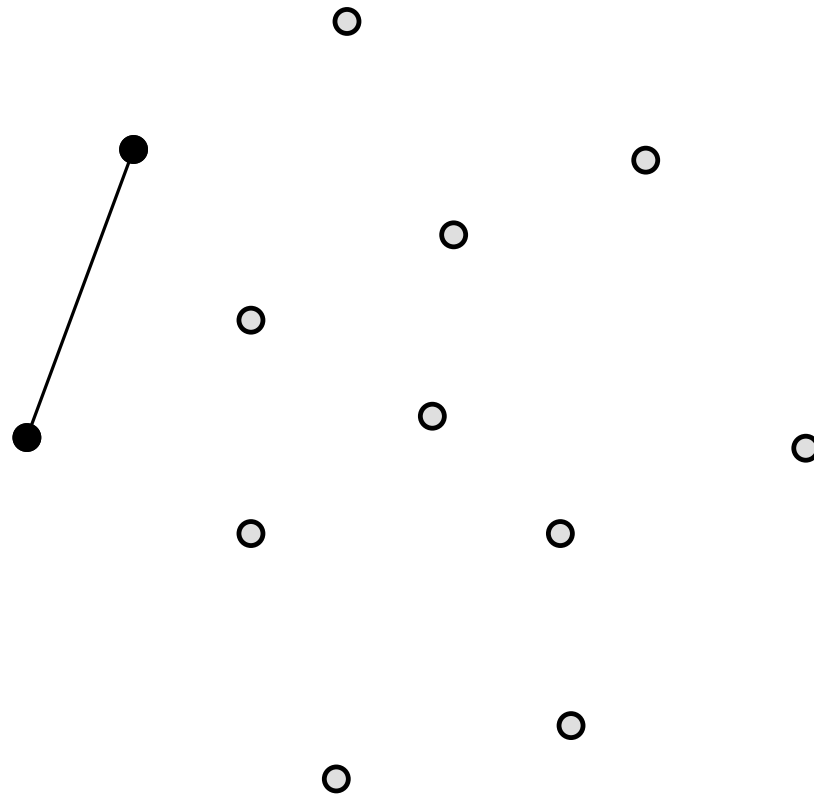
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

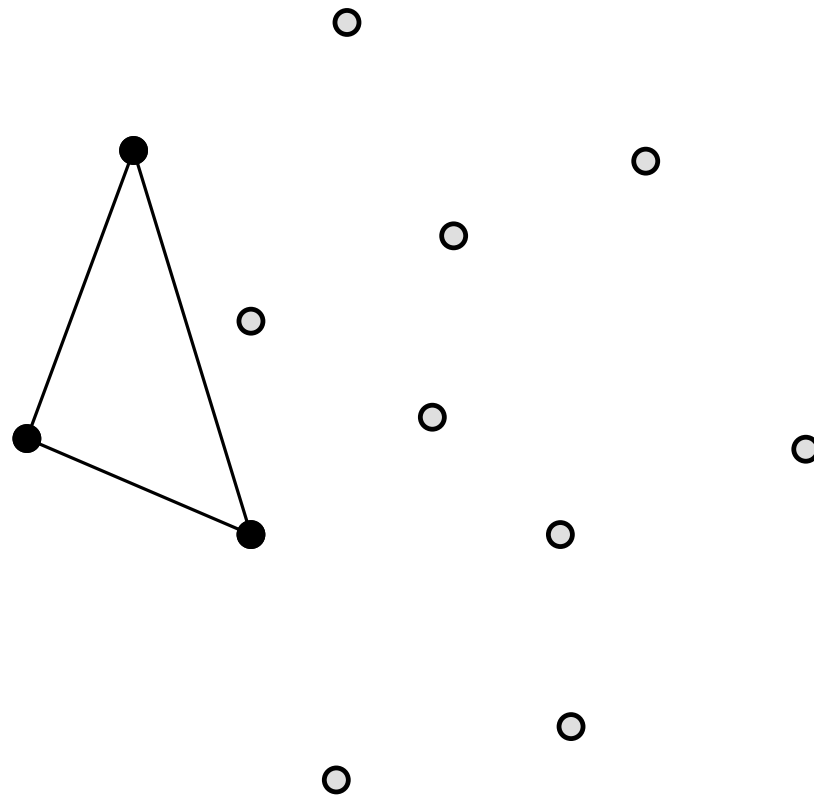
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

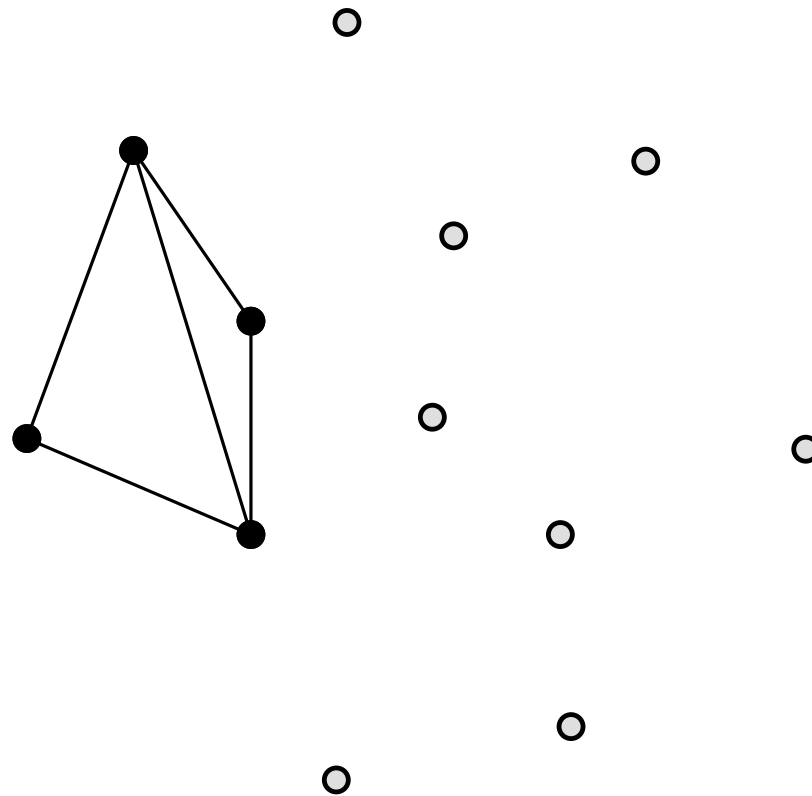
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

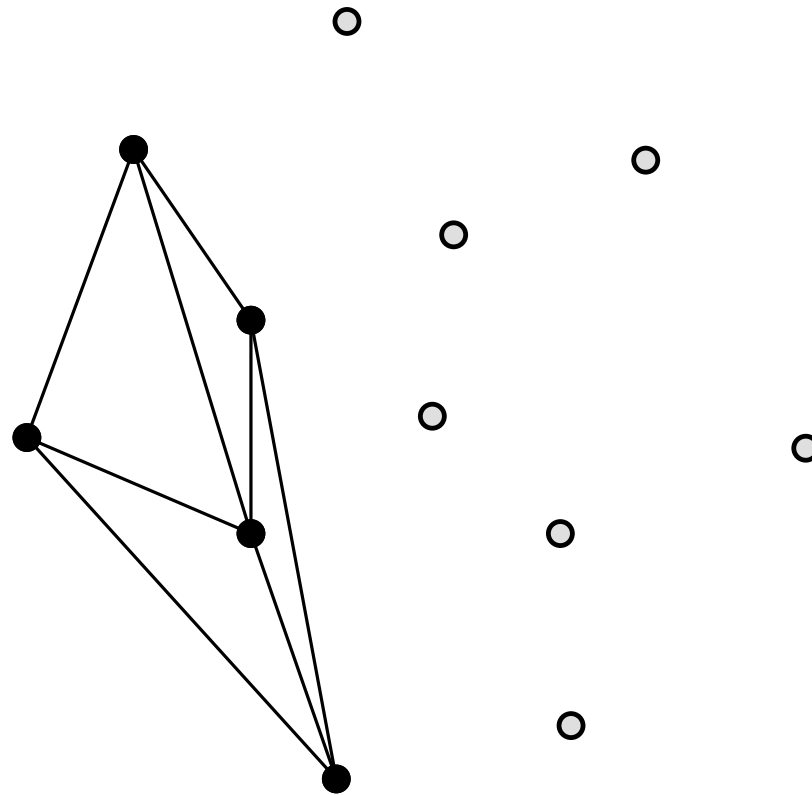
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

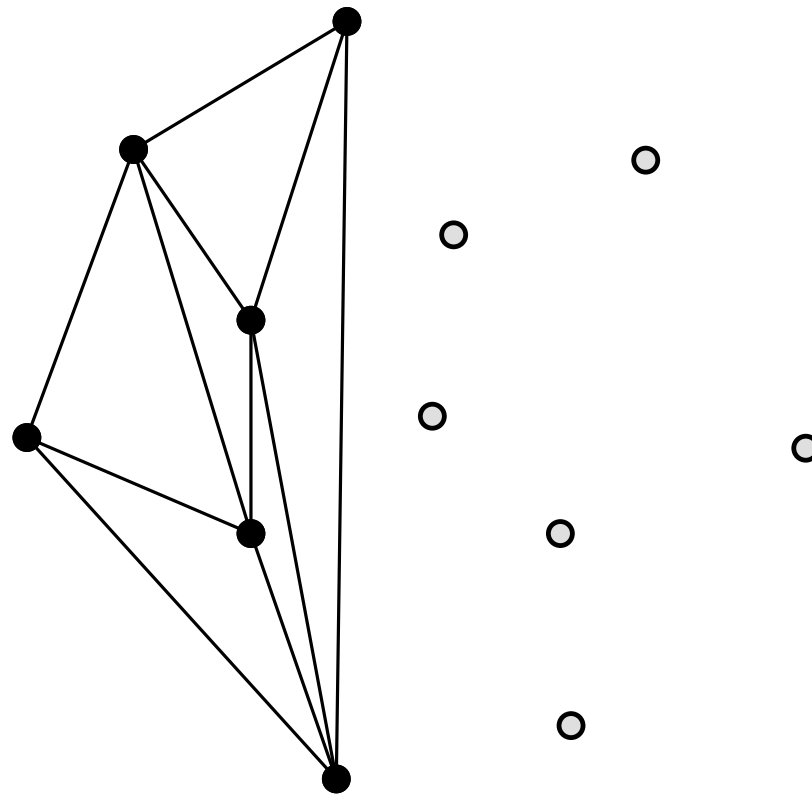
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

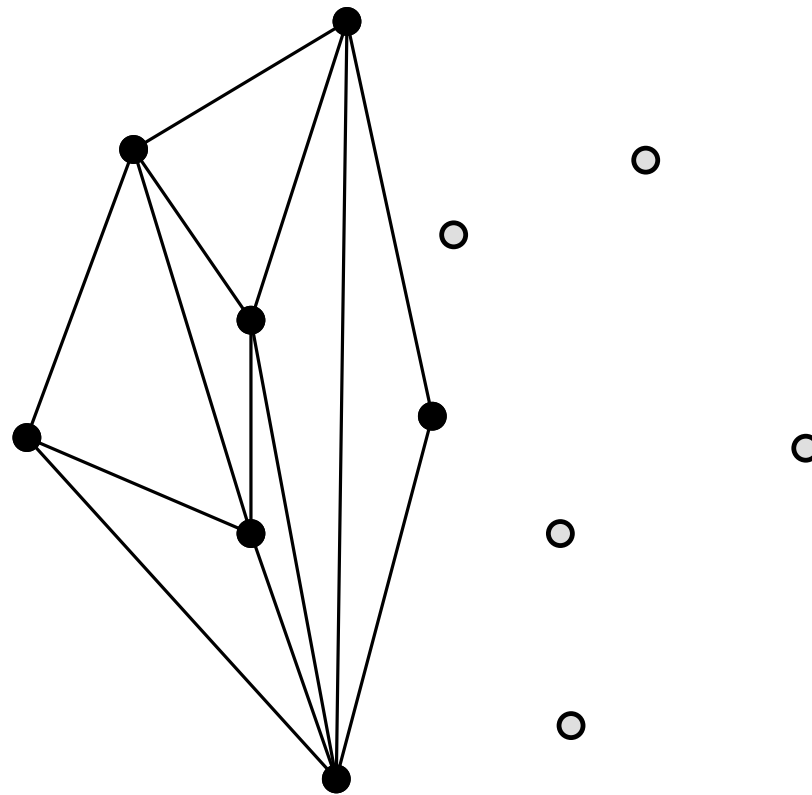
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

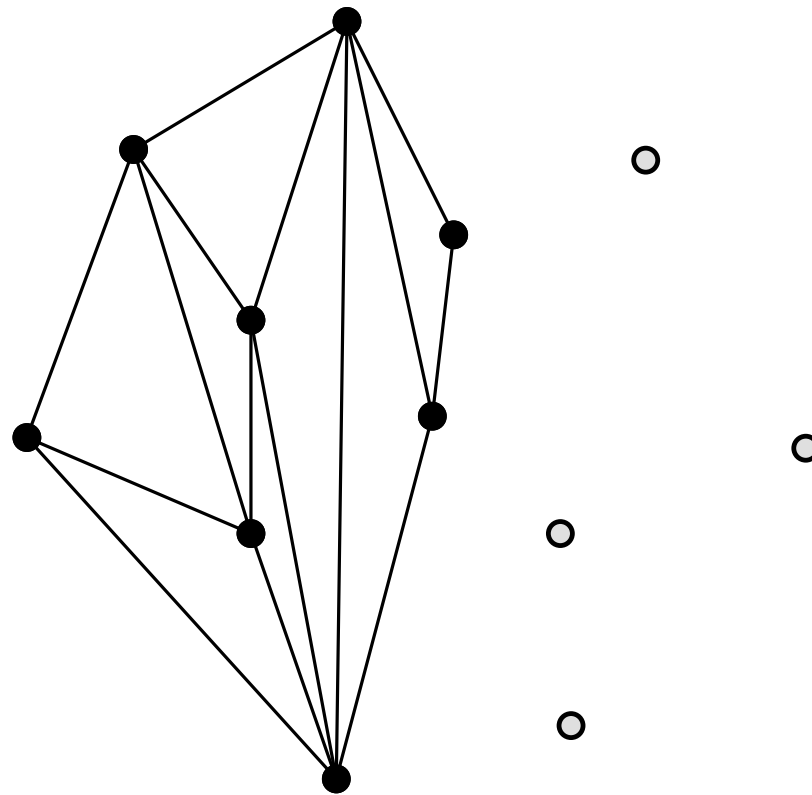
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

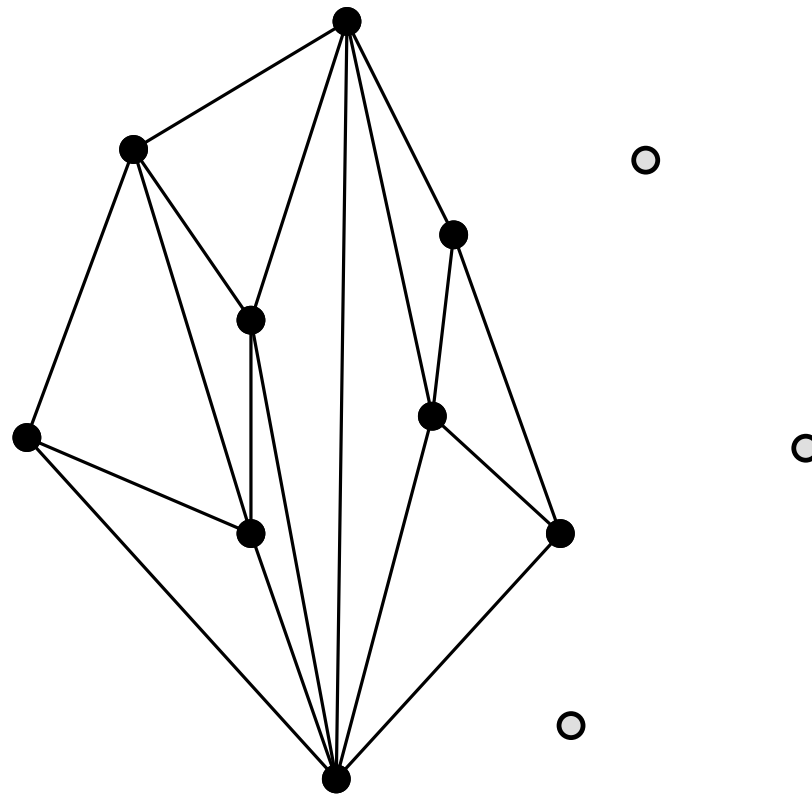
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

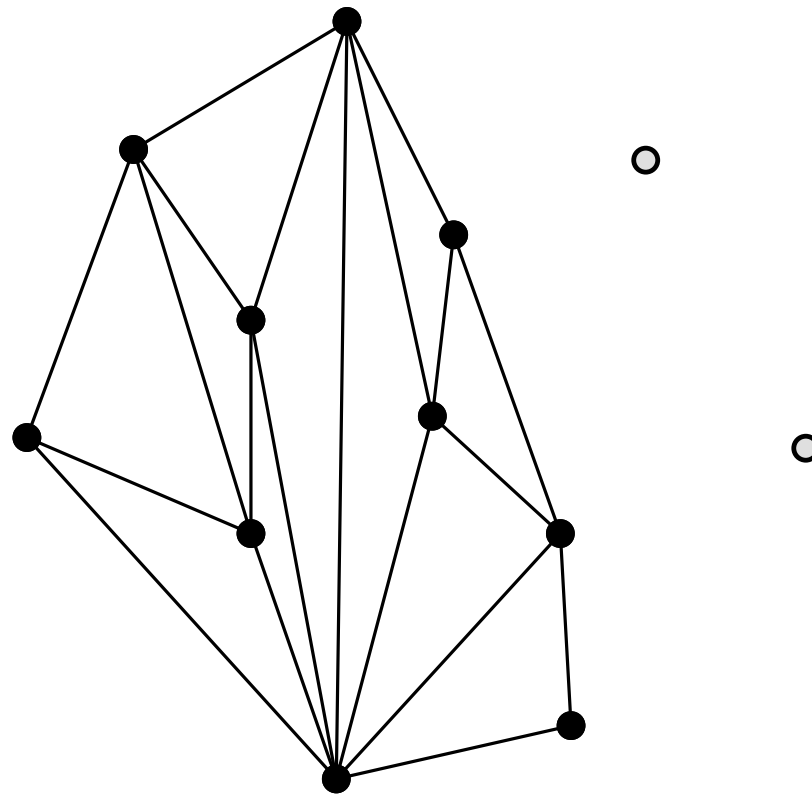
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

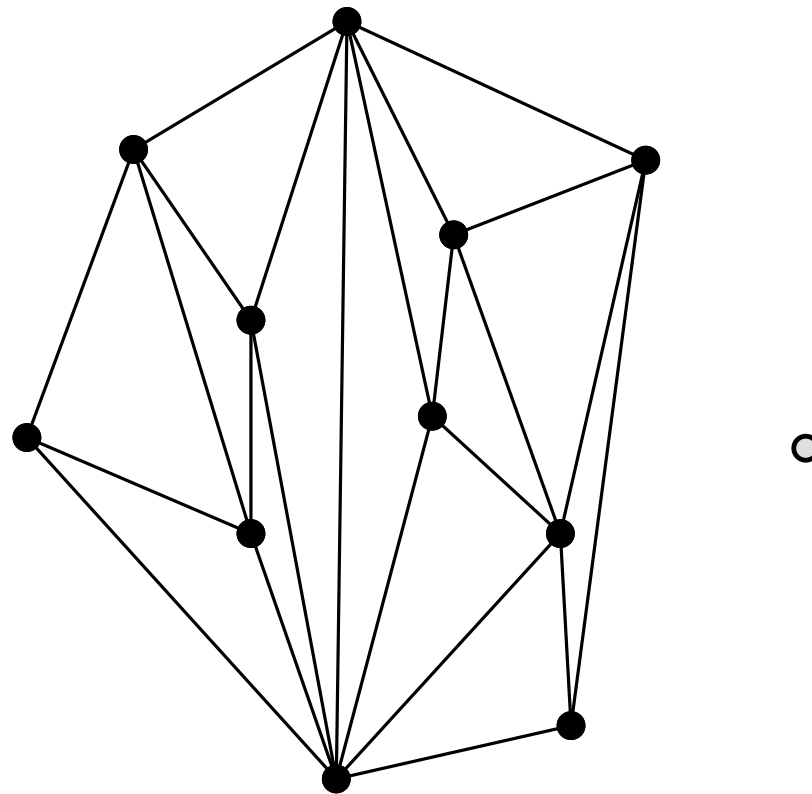
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

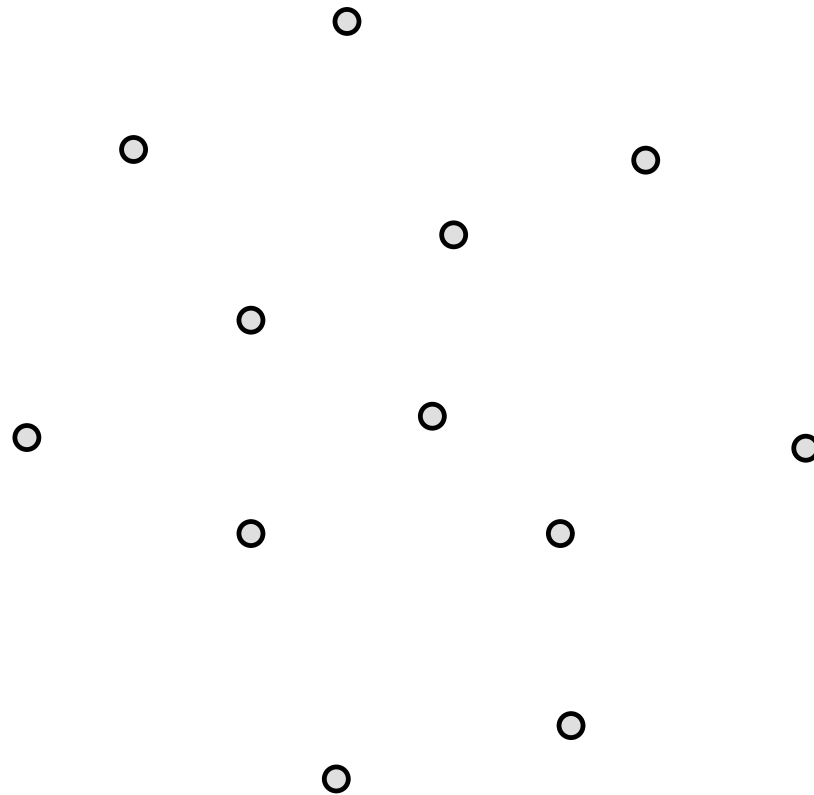
Incremental, sorting



TRIANGULATING POINT SETS

Quality of a triangulation

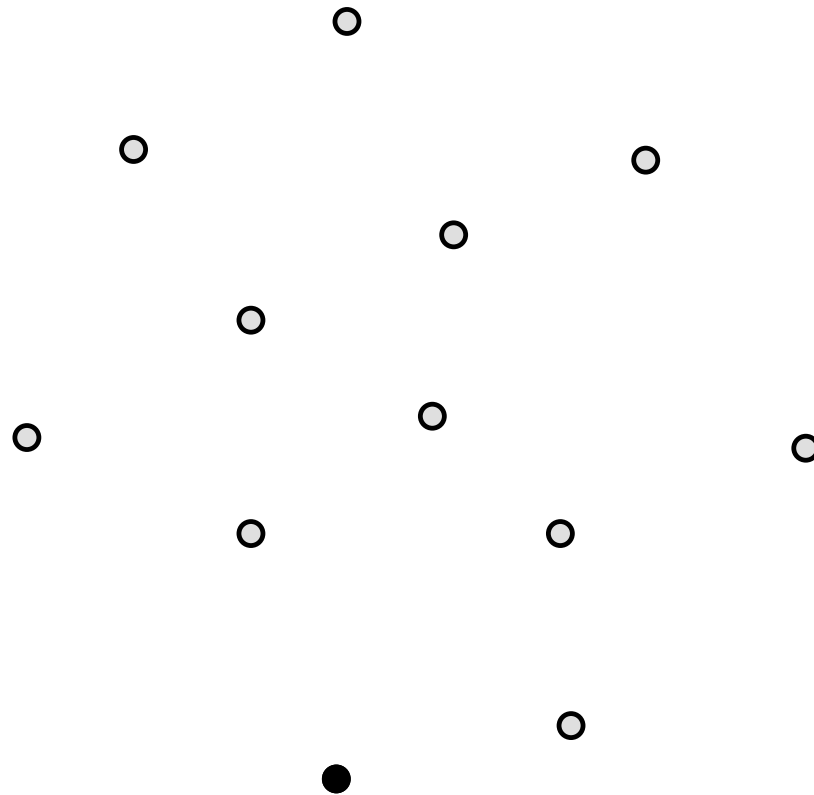
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

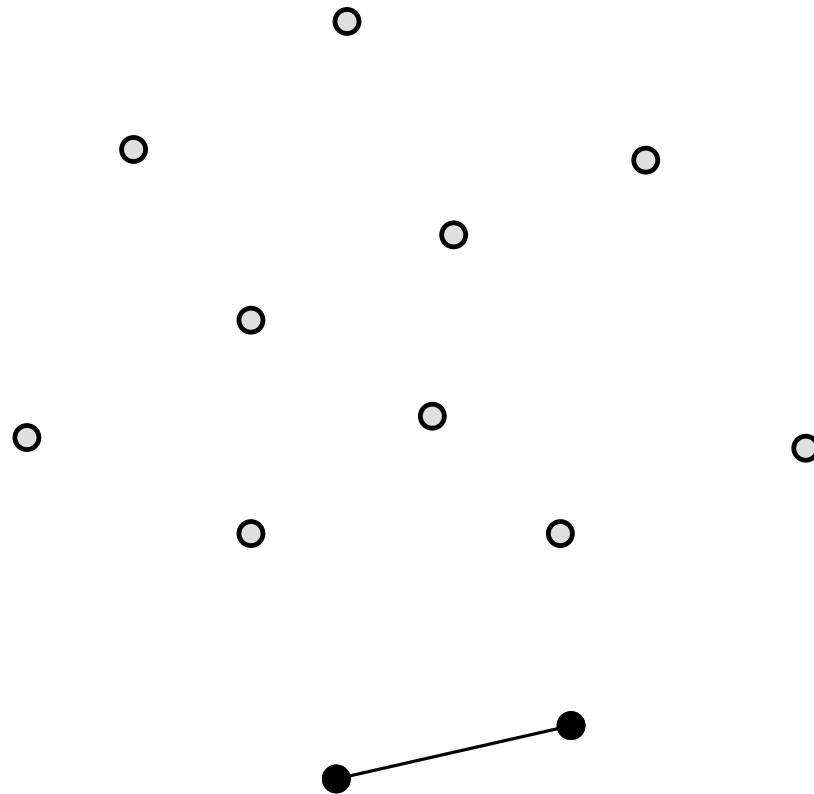
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

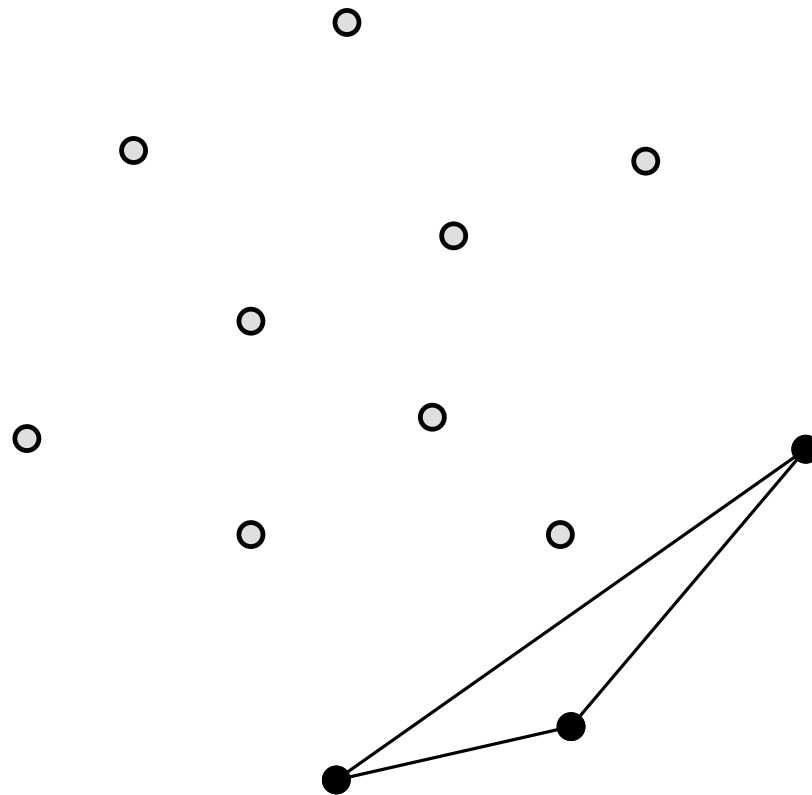
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

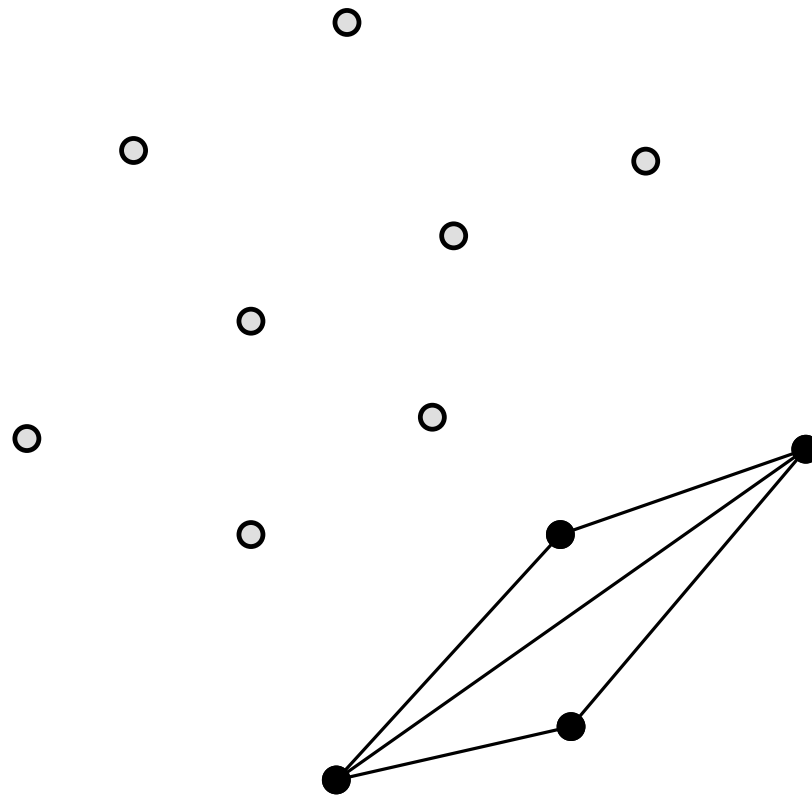
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

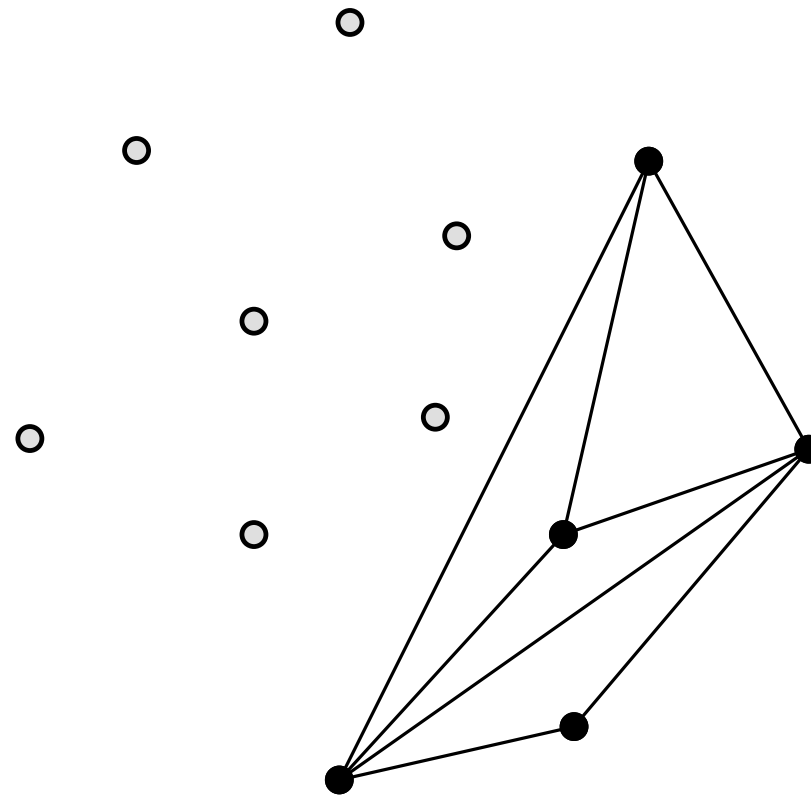
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

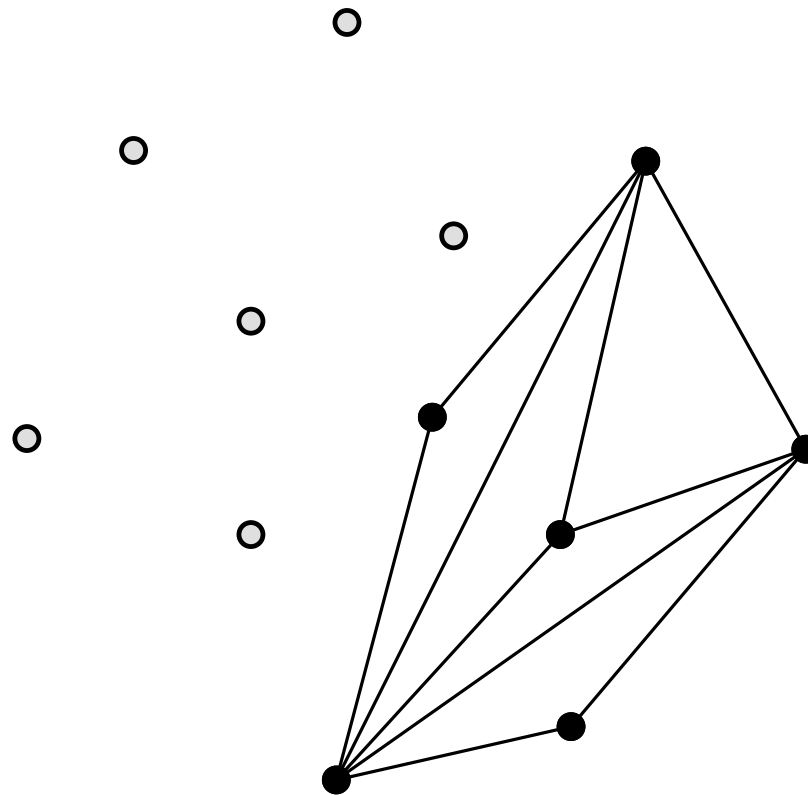
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

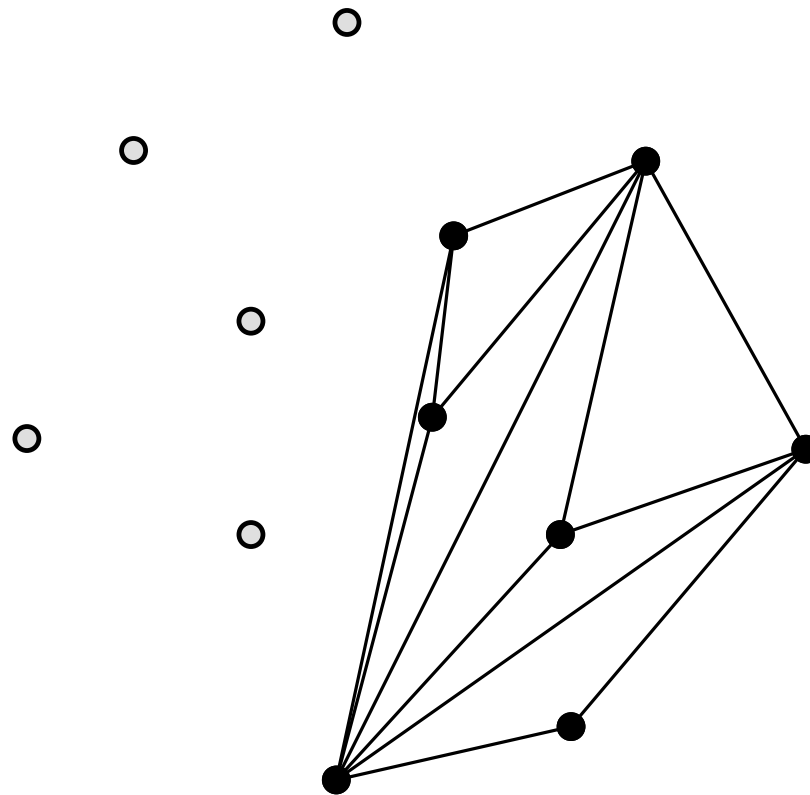
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

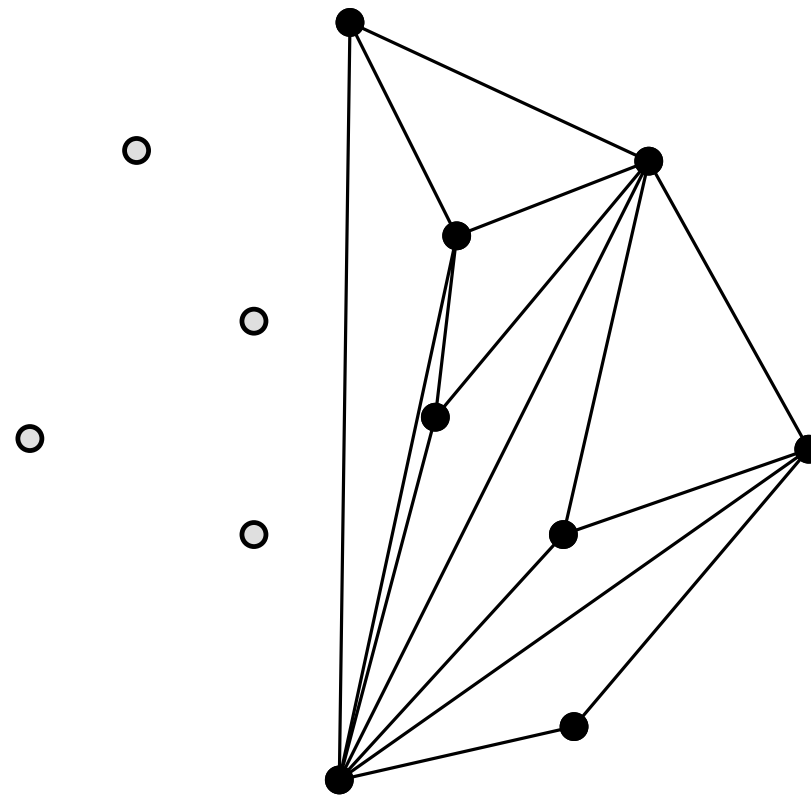
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

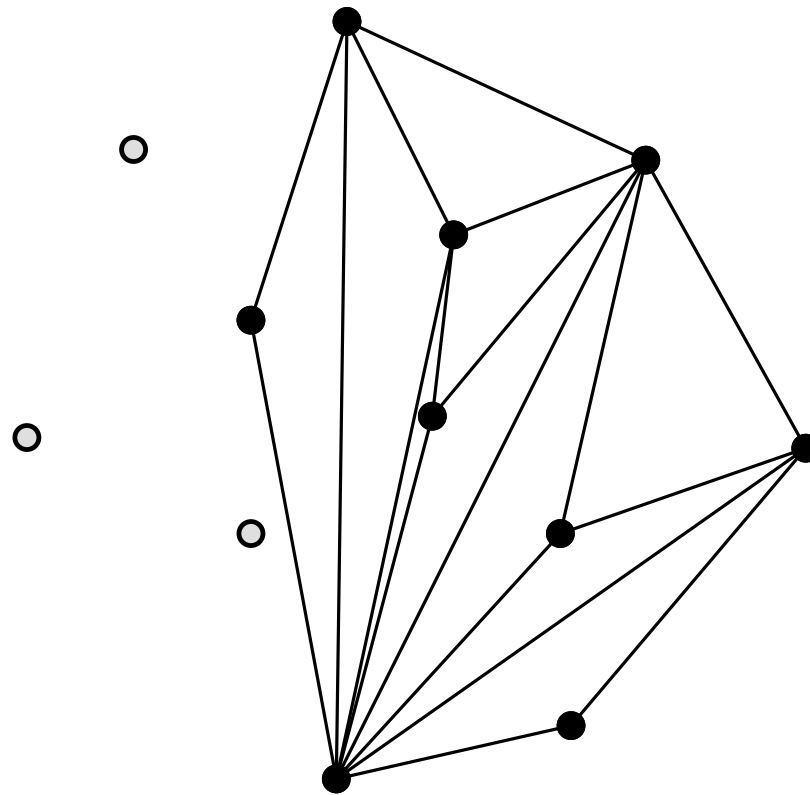
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

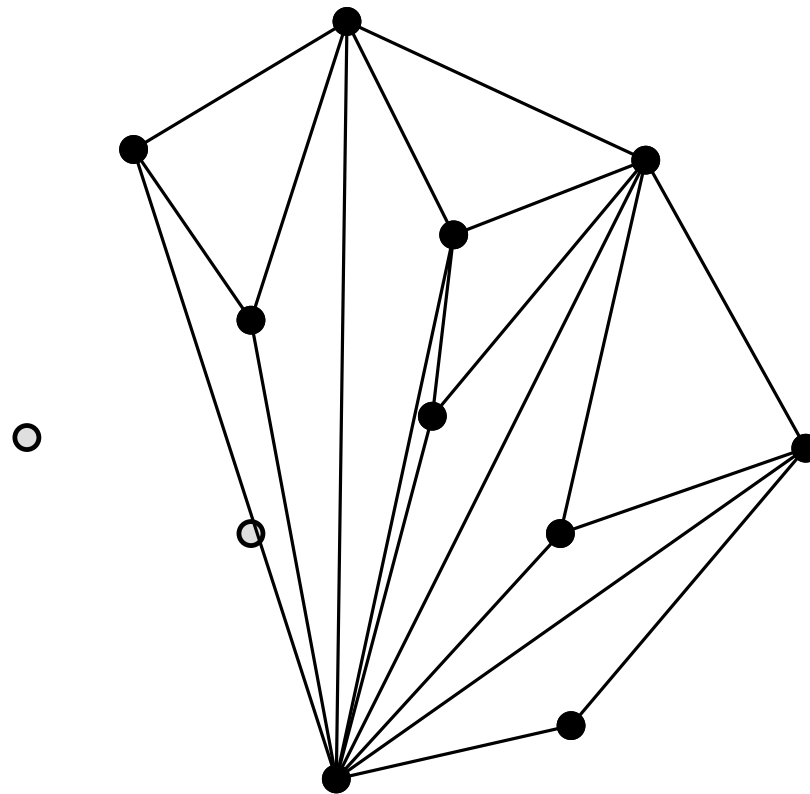
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

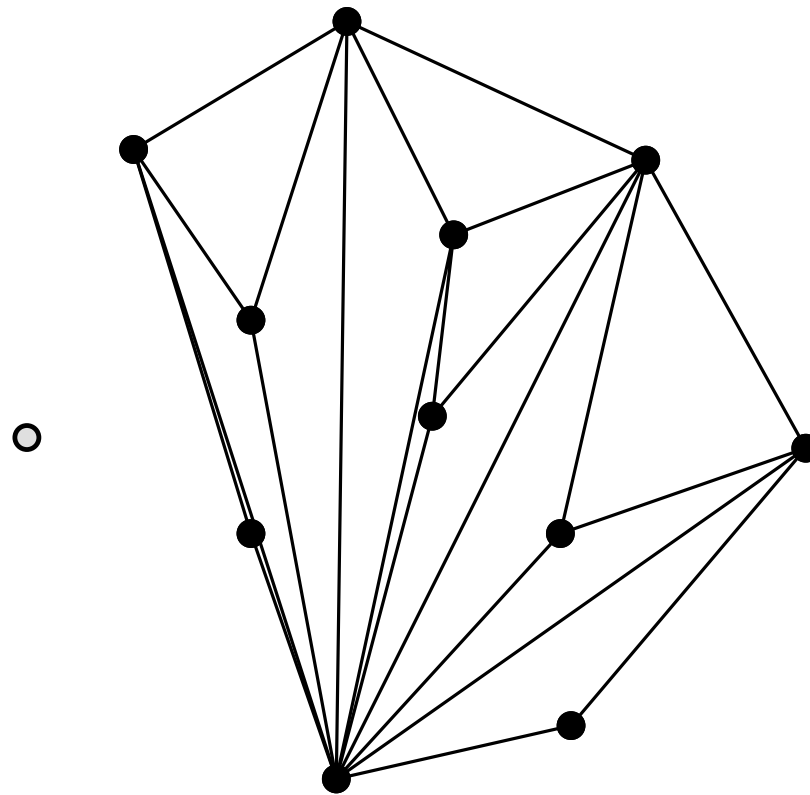
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

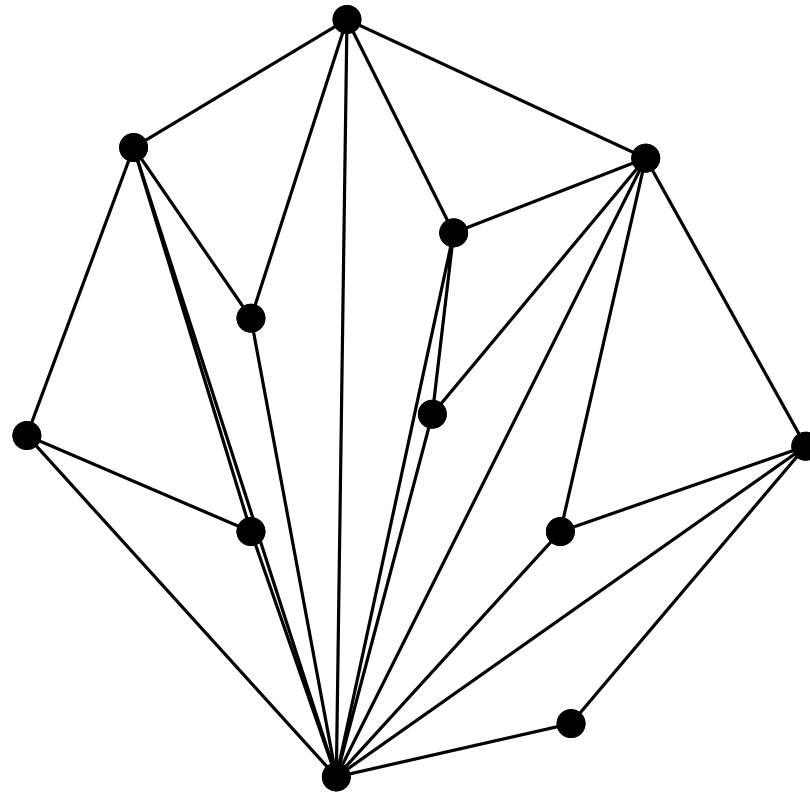
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

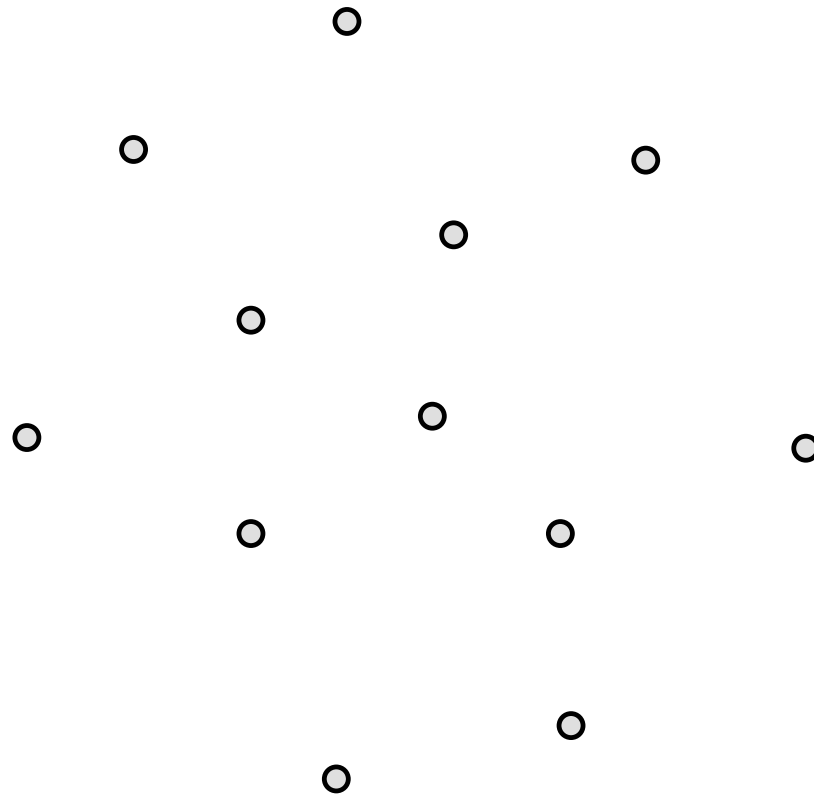
Graham's



TRIANGULATING POINT SETS

Quality of a triangulation

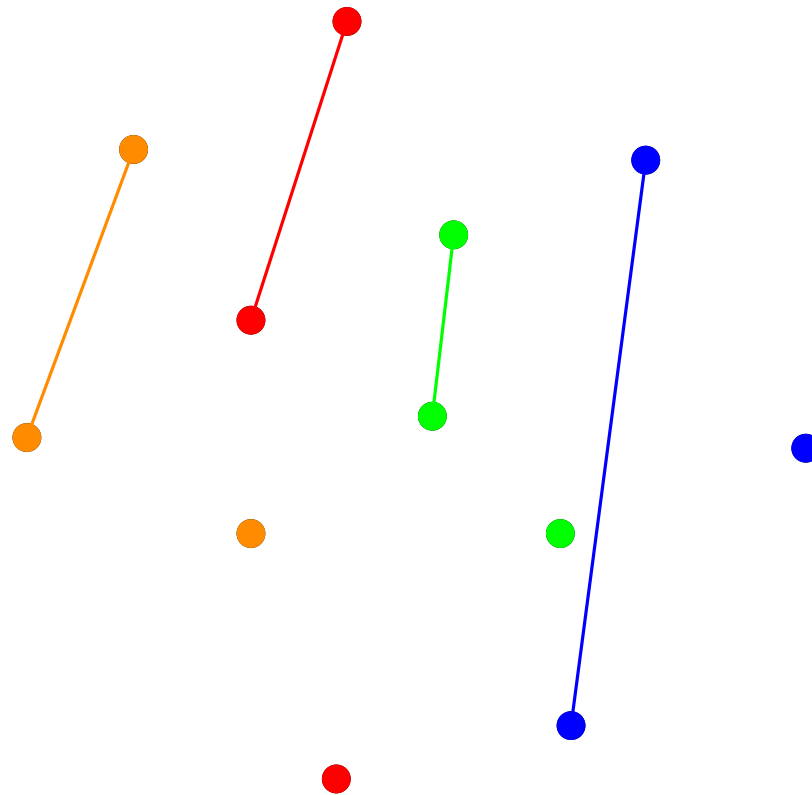
Divide and conquer



TRIANGULATING POINT SETS

Quality of a triangulation

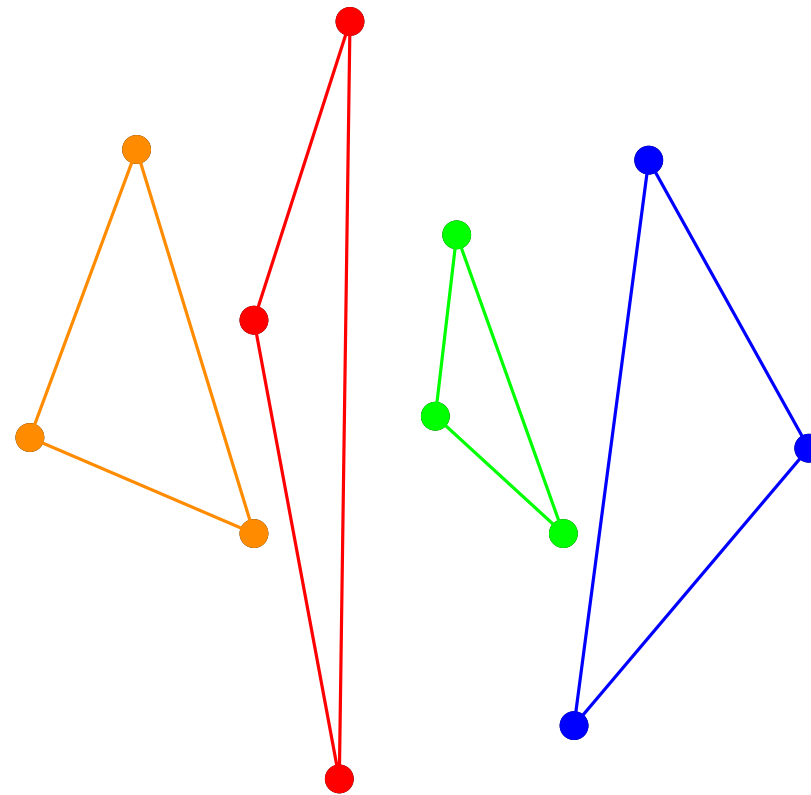
Divide and conquer



TRIANGULATING POINT SETS

Quality of a triangulation

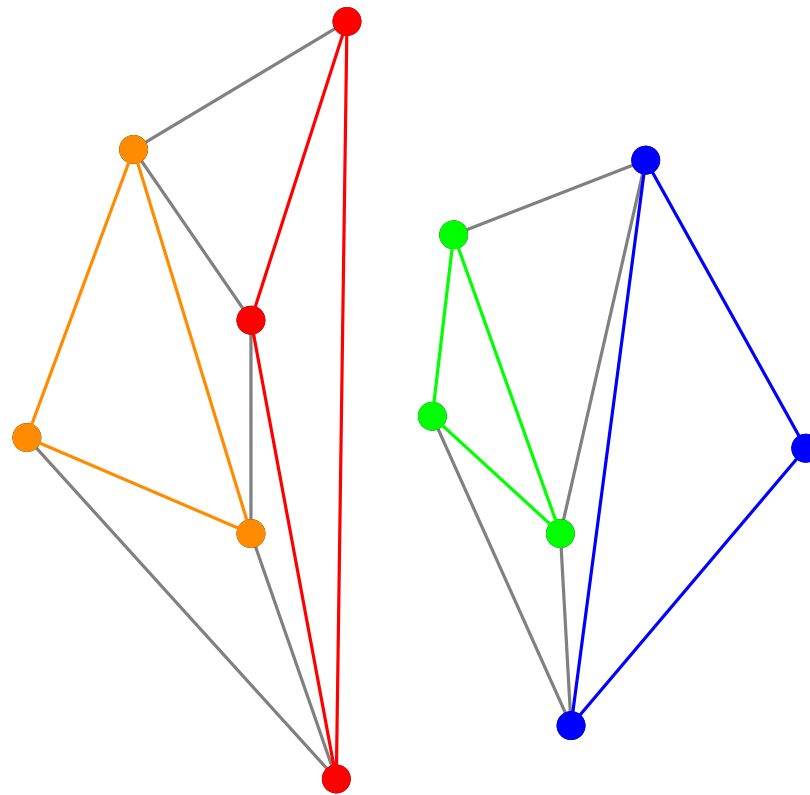
Divide and conquer



TRIANGULATING POINT SETS

Quality of a triangulation

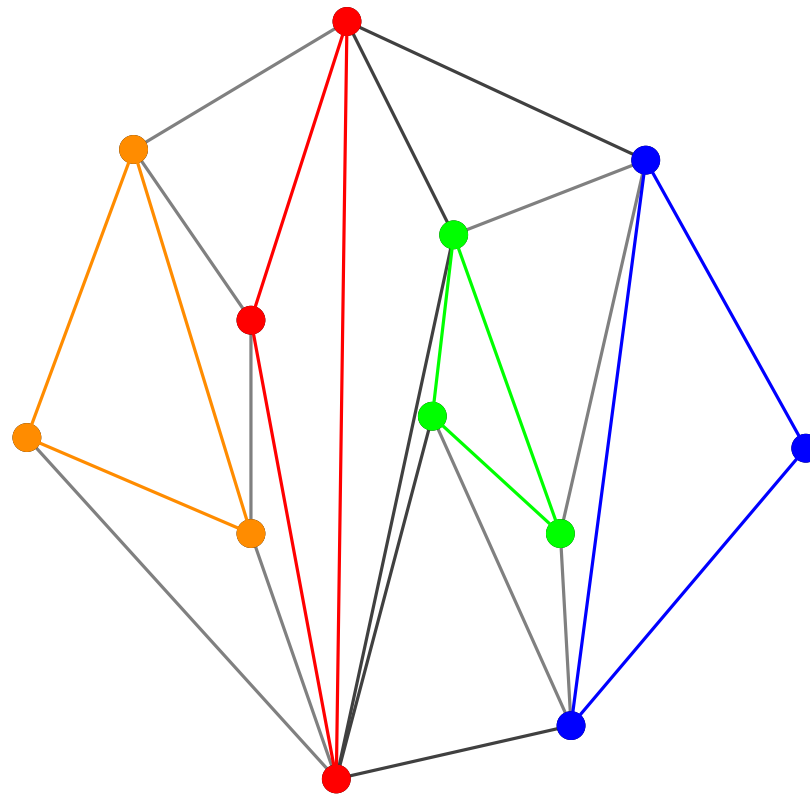
Divide and conquer



TRIANGULATING POINT SETS

Quality of a triangulation

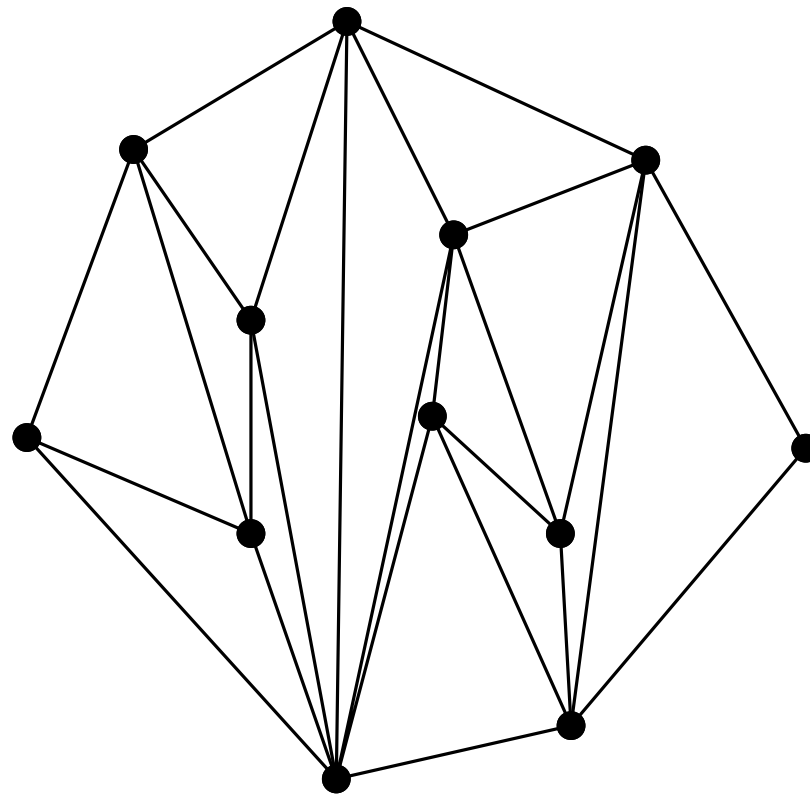
Divide and conquer



TRIANGULATING POINT SETS

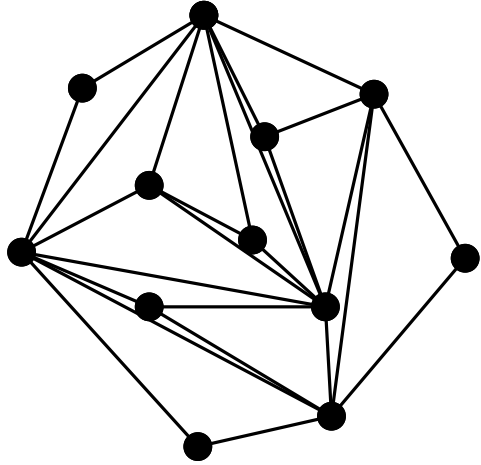
Quality of a triangulation

Divide and conquer

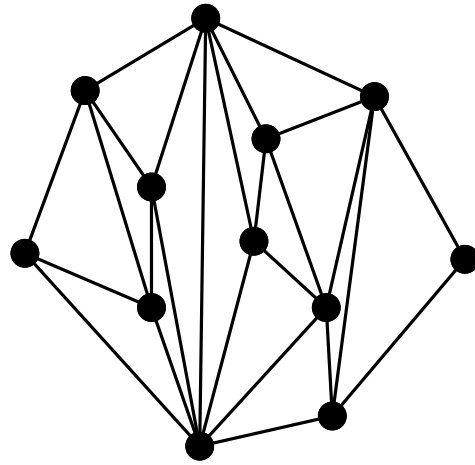


TRIANGULATING POINT SETS

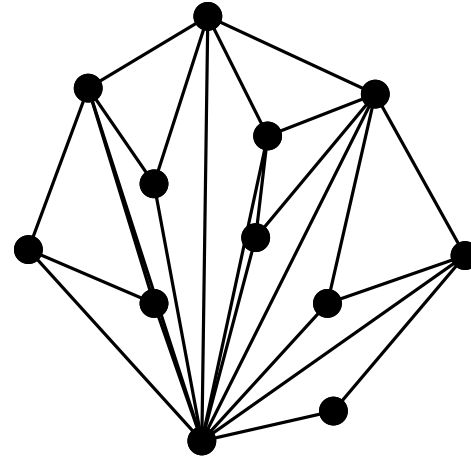
Quality of a triangulation



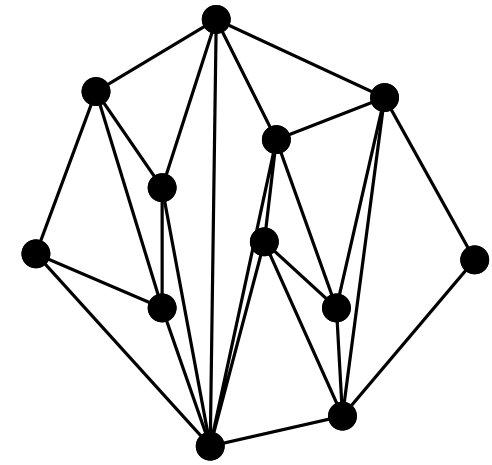
Incremental, unsorted



Incremental, sorted



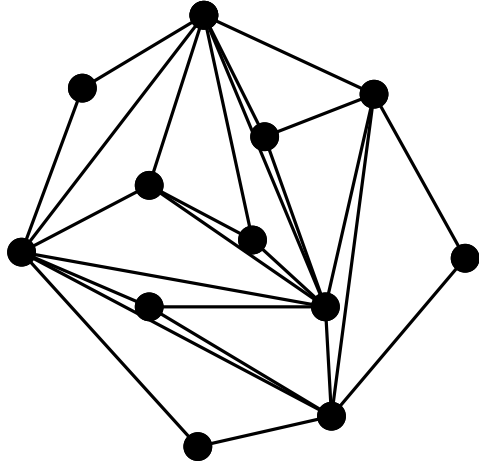
Graham's scan



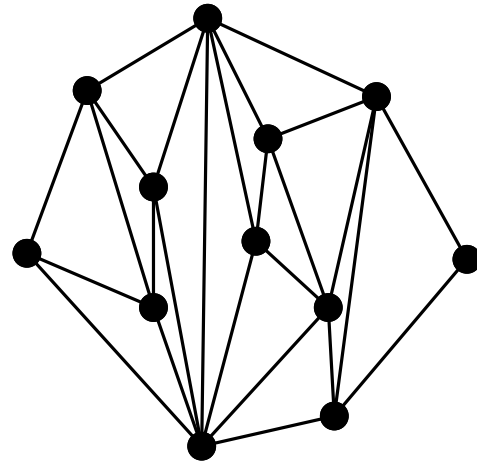
Divide and conquer

TRIANGULATING POINT SETS

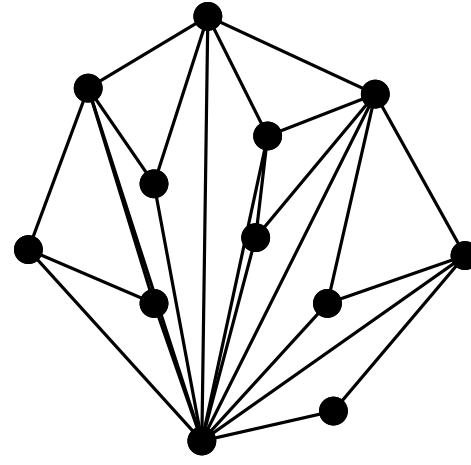
Quality of a triangulation



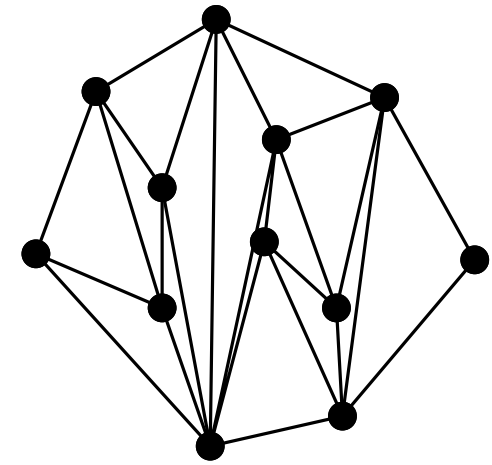
Incremental, unsorted



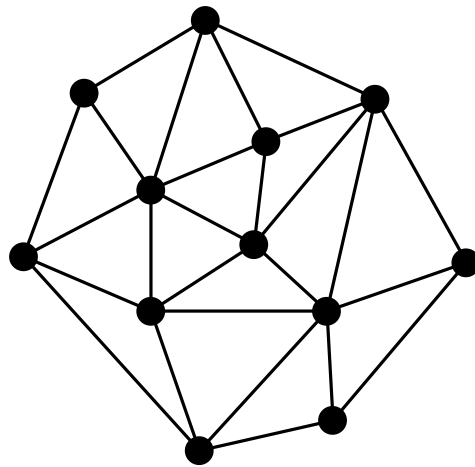
Incremental, sorted



Graham's scan



Divide and conquer

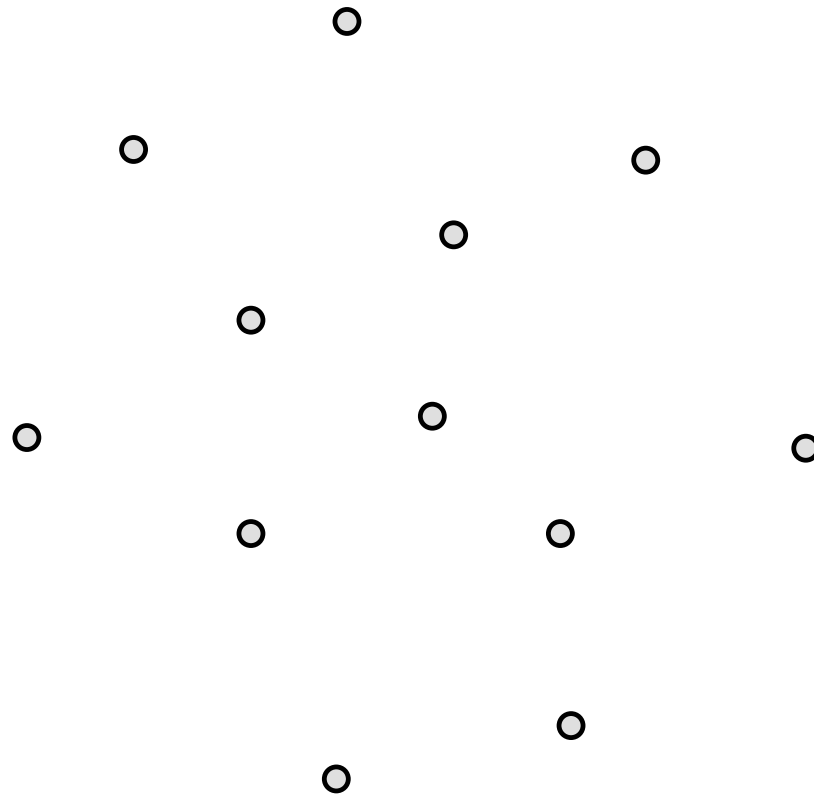


Delaunay triangulation

TRIANGULATING POINT SETS

Quality of a triangulation

Delaunay



TRIANGULATING POINT SETS

Quality of a triangulation

Delaunay

